

AD-A156 856

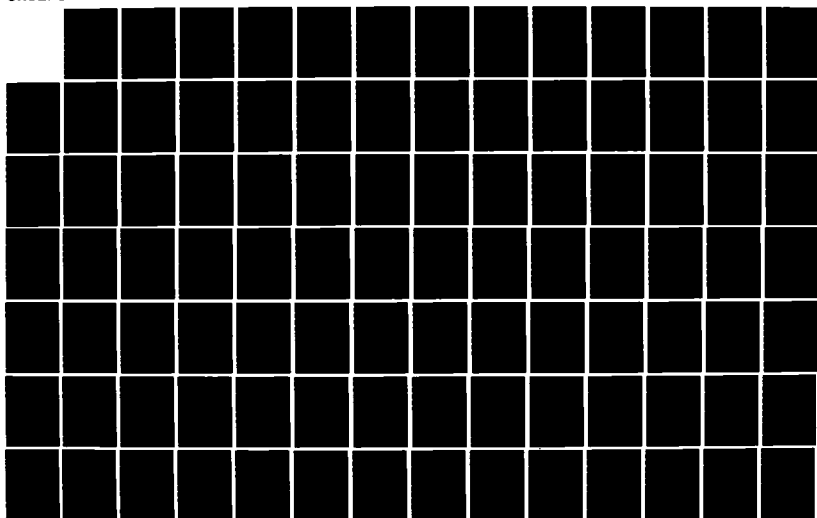
SUPERVISORY CONTROL OF THE RIGHT ARM OF THE BEAM
ASSEMBLY TELEOPERATOR(U) ARMY MILITARY PERSONNEL CENTER
ALEXANDRIA VA A J MANGANIello 10 MAY 85

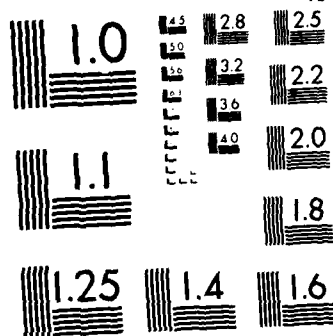
1/2

UNCLASSIFIED

F/G 13/9

NL





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

②

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

AD-A156 856

4. TITLE (and Subtitle)

SUPERVISORY CONTROL OF THE RIGHT ARM OF
THE BEAM ASSEMBLY TELEOPERATOR

5. TYPE OF REPORT & PERIOD COVERED

Final Report May 85

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

CPT. Anthony J. Manganiello, U.S. Army

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Student

HQDA, MILPERCEN, (DAPC-OPA-E)
200 Stovall Street
Alexandria, VA 22332

CONTROLLING OFFICE NAME AND ADDRESS

HQDA, MILPERCEN

ATTN: DAPC-OPA-E

200 Stovall Street

Alexandria, Virginia 22332

MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

10. PROGRAM ELEMENT, PROJECT, TASK
AREA & WORK UNIT NUMBERS

12. REPORT DATE

10 May 85

13. NUMBER OF PAGES

171

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING
SCHEDULE

DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC

ELECTE

JUL 13 1985

18. SUPPLEMENTARY NOTES

Thesis submitted in partial fulfillment for a Master's
of Science Degree in Mechanical Engineering at MIT.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Supervisory Control
Control of a Manipulator
Teleoperator

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis develops a method of supervisory control of the
five-degree-of-freedom dexterous right arm of the Beam Assembly
Teleoperator (BAT).The thesis imposes a constraint that the end effector of the
right arm must move in a straight line. The joint angles are
used to transform the position of the end effector into

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

85-06-18-100

AD-A156 856

DTIC FILE COPY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(When Data Entered)

cylindrical coordinates. With the initial and final position known, an equation for a straight line is obtained and the cylindrical coordinates of the end effector for each point on the straight line are found. The cylindrical coordinates are then transformed back into joint angles for each point and the joint angles are used to control the BAT'S right arm.

The method of moving the end effector in a straight line is implemented in two ways. The first implementation uses an unknown starting position and moves to a known finish point. Here the previously developed BAT software is used to input the starting position of the arm. Then the straight line path method uses this starting point and drives the arm to the known finish point. In the second implementation, a method is developed in which a set of points is remembered by the BAT and the straight line path method is used to drive the arm from any start point, through all the stored points and return to the original point. Further extensions using this method are discussed.

An appendix contains a block diagram description of the Beam Assembly Teleoperator and the source code for the software developed and implemented.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

SUPERVISORY CONTROL OF THE RIGHT ARM OF THE BEAM ASSEMBLY
TELEOPERATOR

CPT. ANTHONY J. MANGANIELLO
HQDA, MILPERCEN (DAPC-OPA-E)
200 Stovall Street
Alexandria, VA 22332

Final Report 10 May 85

Approved for public release; distribution is unlimited.

A thesis submitted to MIT, Cambridge, MA in partial fulfillment
of the requirements for the degree of Master of Science in
Mechanical Engineering.



A1

SUPERVISORY CONTROL OF THE RIGHT ARM
OF THE BEAM ASSEMBLY TELEOPERATOR

by

ANTHONY J. MANGANIELLO

B.S., United States Military Academy
(1977)

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the
Requirements for the
Degree of

MASTER OF SCIENCE in MECHANICAL ENGINEERING

at the

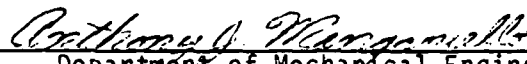
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1985

© Anthony J. Manganiello 1985

The author hereby grants to M.I.T. permission to reproduce
and to distribute copies of this thesis document in whole
or in part.

Signature of Author



Department of Mechanical Engineering
May 10, 1985

Certified by



David L. Akin
Thesis Supervisor

Accepted by



Ain A. Sonin
Chairman, Mechanical Engineering Department Committee

SUPERVISORY CONTROL OF THE RIGHT ARM
OF THE BEAM ASSEMBLY TELEOPERATOR

by

Anthony J. Manganiello

Submitted to the Department of Mechanical Engineering
on May 10, 1985 in partial fulfillment of the
requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

This thesis develops a method of supervisory control of the five-degree-of-freedom dexterous right arm of the Beam Assembly Teleoperator (BAT).

The thesis imposes a constraint that the end effector of the right arm must move in a straight line. The joint angles are used to transform the position of the end effector into cylindrical coordinates. With the initial and final position known, an equation for a straight line is obtained and the cylindrical coordinates of the end effector for each point on the straight line are found. The cylindrical coordinates are then transformed back into joint angles for each point and the joint angles are used to control the BAT'S right arm.

The method of moving the end effector in a straight line is implemented in two ways. The first implementation uses an unknown starting position and moves to a known finish point. Here the previously developed BAT software is used to input the starting position of the arm. Then the straight line path method uses this starting point and drives the arm to the known finish point. In the second implementation, a method is developed in which a set of points is remembered by the BAT and the straight line path method is used to drive the arm from any start point, through all the stored points and return to the original point. Further extensions using this method are discussed.

An appendix contains a block diagram description of the Beam Assembly Teleoperator and the source code for the software developed and implemented.

Thesis Supervisor: David L. Akin
Title: Rockwell International Assistant Professor of
Aeronautics

ACKNOWLEDGEMENTS

I would like to sincerely thank Professor David Akin for all the assistance he has given me. He was always available to help guide and educate me throughout the entire thesis. It was obvious that educating and helping students is his utmost concern.

Very special thanks to John Spofford who helped teach me everything I know about the BAT and never once made me wait when I asked for help. Good luck in the future with all your work, John.

Many thanks also go to everyone in the Space System Laboratory including those members now gone who have worked on the BAT project from its conception (in particular, Tom Waschura for his work on the software and Eric Shain for his work on the wrist). If it were not for their hard work and education this thesis would have never been conceived.

Thanks to Homayoon Kazerooni for his guidance concerning M.I.T. and the Mechanical Engineering Department. Thanks for taking me under your wing and helping me.

I would also like to thank Sandy Tepper for not only typing my thesis and helping me out from the start of my stay at M.I.T., but also for knowing more about M.I.T. than anyone else.

I would like to thank the U.S. Army for making it possible for me to obtain this degree.

Most of all, I would like to thank my wife, Susan, and my children, Christopher and Lisa, for their patience, understanding and support. I only hope I can repay them for the sacrifices they made for me while I was at M.I.T. Thanks for always being there.

TABLE OF CONTENTS

	PAGE
ABSTRACT	2
ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	5
LIST OF FIGURES	6
CHAPTER 1 - INTRODUCTION	11
CHAPTER 2 - A METHOD FOR SUPERVISORY CONTROL OF THE RIGHT ARM	15
2.1 Objective	15
2.2 Overview of the Procedure	17
2.3 Problem Set-Up	19
2.4 Joint Angles to Cylindrical Coordinates	23
2.5 Straight Line Path Generation	29
2.6 Cylindrical Coordinates to Joint Angles	36
2.7 Determination of the Total Time and the Values of the Shoulder Yaw (θ_1) at Each Time Step	41
2.8 Determination of the Orientation Angles	51
2.9 Simulation	52
CHAPTER 3 - IMPLEMENTATION I : UNKNOWN STARTING POSITION, KNOWN FINISH POINT	71
CHAPTER 4 - IMPLEMENTATION II: LEARNING TRAJECTORIES BY CONCATENATION	89
CHAPTER 5 - CONCLUSIONS	103
REFERENCES	107
APPENDIX A - DESCRIPTION OF THE BAT	108
APPENDIX B - PROGRAM FOR SUPERVISORY CONTROL	147

LIST OF FIGURES

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
1.1	Sketch of the ICS and BAT	12
2.1	Diagram of the Right Arm Control System	16
2.2	Joint Angles	19
2.3	Point A on the Right Arm	20
2.4	Coordinate Frames for the Right Arm	21
2.5	Coordinate Frames for the Wrist	21
2.6	Cylindrical Coordinate Reference Frame	23
2.7	Joint Angles to Cylindrical Coordinates - Position i	24
2.8	Joint Angles to Cylindrical Coordinates - Position ii	26
2.9	Joint Angles to Cylindrical Coordinates - Position iii	28
2.10	Straight Line Path in Reference Frame (X_0, Y_0, Z_0)	30
2.11	Projected Line in X_0Y_0 Plane	30
2.12	Travelling Across the Projected Line	31
2.13	Slope of the Projected Line Equal to 1	35
2.14	Cylindrical Coordinates to Joint Angles - Case i	37
2.15	Cylindrical Coordinates to Joint Angles - Case ii	39
2.16	Cylindrical Coordinates to Joint Angles - Case iii	41
2.17a	Driving Function (Initial Less than Final)	42
2.17b	Driving Function (Initial Greater than Final)	43

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
2.18	Simplified Model of Arm and Beam	46
2.19	Sampled Driving Function	49
2.20	Arm Position - Simulation I	55
2.21	Shoulder Yaw Position - Simulation I	56
2.22	Shoulder Yaw Velocity - Simulation I	56
2.23	Shoulder Yaw Acceleration - Simulation I	57
2.24	Wrist Yaw Position - Simulation I	57
2.25	Wrist Roll Position - Simulation I	58
2.26	Y vs. X Position of Point A - Simulation I	58
2.27	Z vs. X Position of Point A - Simulation I	59
2.28	Z vs. Y Position of Point A - Simulation I	59
2.29	Arm Position - Simulation II	60
2.30	Shoulder Yaw Position - Simulation II	61
2.31	Shoulder Yaw Velocity - Simulation II	61
2.32	Shoulder Yaw Acceleration - Simulation II	62
2.33	Wrist Yaw Position - Simulation II	62
2.34	Wrist Roll Position - Simulation II	63
2.35	Y vs. X Position of Point A - Simulation II	63
2.36	Z vs. X Position of Point A - Simulation II	64
2.37	Z vs. Y Position of Point A - Simulation II	64
2.38	Arm Position - Simulation III	65
2.39	Shoulder Yaw Position - Simulation III	66

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
2.40	Arm Radius Velocity - Simulation III	66
2.41	Arm Radius Acceleration - Simulation III	67
2.42	Wrist Yaw Position - Simulation III	67
2.43	Wrist Roll Position - Simulation III	68
2.44	Y vs. X Position of Point A - Simulation III	68
2.45	Z vs. X Position of Point A - Simulation III	69
2.46	Z vs. Y Position of Point A - Simulation III	69
3.1	Finish Point, Implementation I	72
3.2	Arm Position, Step 1, Implementation I	81
3.3	Shoulder Yaw Position, Step 1, Implementation I	82
3.4	Wrist Yaw Position, Step 1, Implementation I	82
3.5	Wrist Roll Position, Step 1, Implementation I	83
3.6	Y vs. X Position, Step 1, Implementation I	83
3.7	Z vs. X Position, Step 1, Implementation I	84
3.8	Z vs. Y Position, Step 1, Implementation I	84
3.9	Shoulder Yaw Position, Step 2, Implementation I	85
3.10	Wrist Yaw Position, Step 2, Implementation I	86
3.11	Wrist Roll Position, Step 2, Implementation I	86
3.12	Y vs. X Position, Step 2, Implementation I	87
3.13	Z vs. X Position, Step 2, Implementation I	87
3.14	Z vs. Y Position, Step 2, Implementation I	88
4.1	Implementation II	91
4.2	Arm Position, Step 1, Implementation II	95

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
4.3	Shoulder Yaw Position, Step 1, Implementation II	96
4.4	Wrist Yaw Position, Step 1, Implementation II	96
4.5	Wrist Roll Position, Step 1, Implementation II	97
4.6	Y vs. X Position, Step 1, Implementation II	97
4.7	Z vs. X Position, Step 1, Implementation II	98
4.8	Z vs. Y Position, Step 1, Implementation II	98
4.9	Shoulder Yaw Position, Step 2, Implementation II	99
4.10	Wrist Yaw Position, Step 2, Implementation II	100
4.11	Wrist Roll Position, Step 2, Implementation II	100
4.12	Y vs. X Position, Step 2, Implementation II	101
4.13	Z vs. X Position, Step 2, Implementation II	101
4.14	Z vs. Y Position, Step 2, Implementation II	102
A.1	Sketch of BAT	108
A.2	Main Power Subsystem	110
A.3	Control Subsystem	112
A.4	Pneumatic Subsystem	113
A.5	Video Subsystem	115
A.6	Left Arm Subsystem	117
A.7	Right Arm Subsystem	118
A.8	Propulsion Subsystem	120
A.9	Sketch of ICS	121
A.10	ICS Control Panels	123
A.11	Central Controller	126

As stated previously, the concern in this chapter is to develop a method of generating a straight line path for point A. In this chapter the values of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ (shoulder yaw, shoulder pitch, elbow, wrist motor A, wrist motor B) are given as part of the problem. ψ_4 and ψ_5 will be tucked away until Section 2.8 (since the motor angles do not affect the position of point A).

At this time, the starting position of point A is represented in terms of $\theta_1, \theta_2, \theta_3$ (shoulder yaw, shoulder pitch, elbow) and the length of the two links (L_1 and L_2). It would be easier to compute a straight line path if the joint coordinates ($\theta_1, \theta_2, \theta_3, L_1, L_2$) were changed to another coordinate system.

2.4 Joint Angles to Cylindrical Coordinates

Cylindrical coordinates θ, R, Z (Figure 2.6) were chosen because of both the design of the manipulator and the ease of transformation. Shoulder

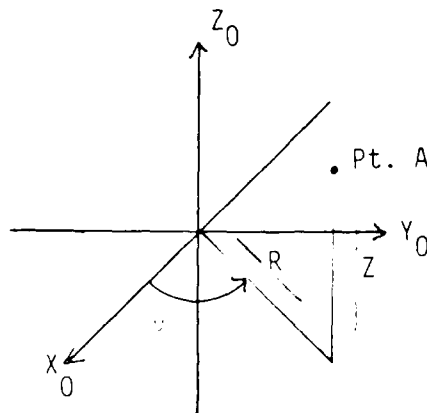


FIGURE 2.6 Cylindrical Coordinate Reference Frame

formation is wrist roll, θ_5 . θ_5 rotates about the Y" axis from $\theta_{5\text{MIN}} = -1.745$ radians to $\theta_{5\text{MAX}} = 1.745$ radians. There is one other consideration concerning the wrist angles. When the wrist was developed by the Space Systems Laboratory, a differential mechanism and two motors (motor A and motor B) were used inside the wrist to enable it to have yaw and roll. A detailed description of the mechanism is described in reference [3]. The angles which are used in the control system are the angles from the wrist motors; these motors are labelled A and B. In this thesis ψ_4 will be used as the angle for motor A and ψ_5 will be used as the angle for motor B. Reference [3] develops the following transformation used to convert between ψ_4, ψ_5 and θ_4, θ_5 (the two motor angles and the two wrist angles).

$$\theta_4 = -\frac{1}{2} \psi_4 + \frac{1}{2} \psi_5$$

$$\theta_5 = -\frac{1}{2a} \theta_4 - \frac{1}{2a} \theta_5$$

where $a = \frac{27}{29}$, and is based on the gear ratios of the differential drive.

Also,

$$\psi_4 = -\theta_4 - a \theta_5$$

$$\psi_5 = \theta_4 - a \theta_5$$

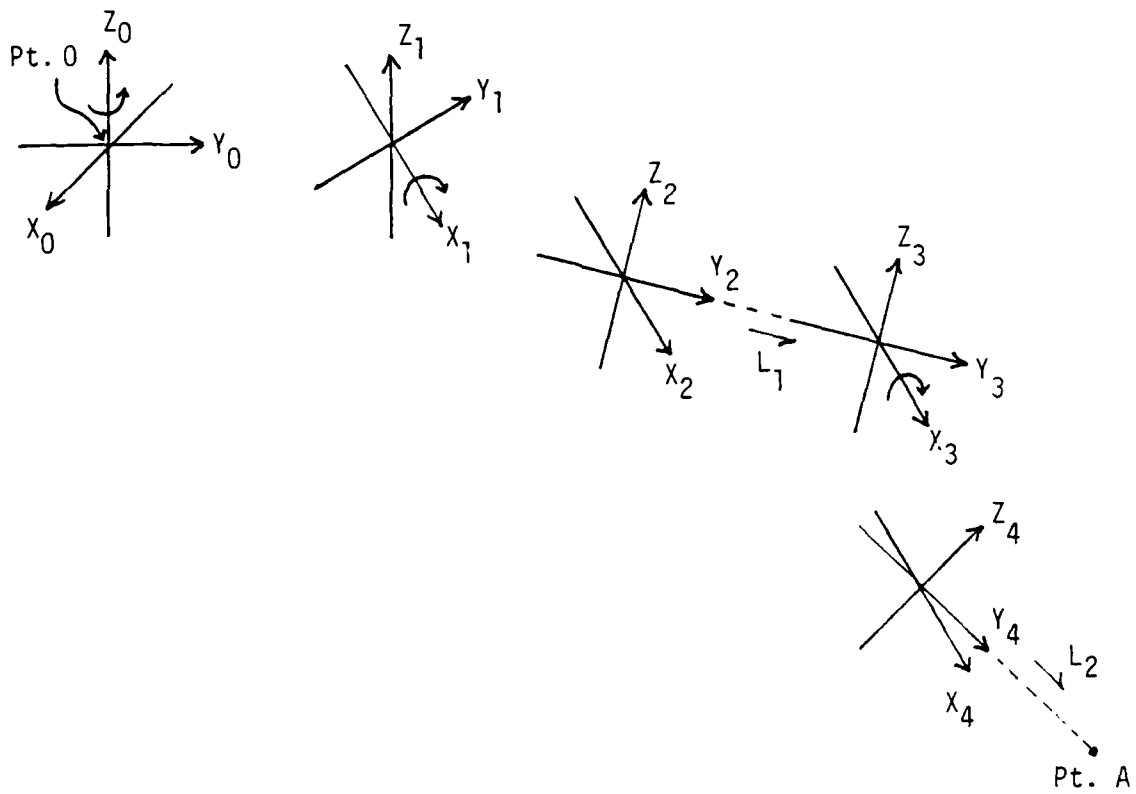


FIGURE 2.4 Coordinate Frames for the Right Arm

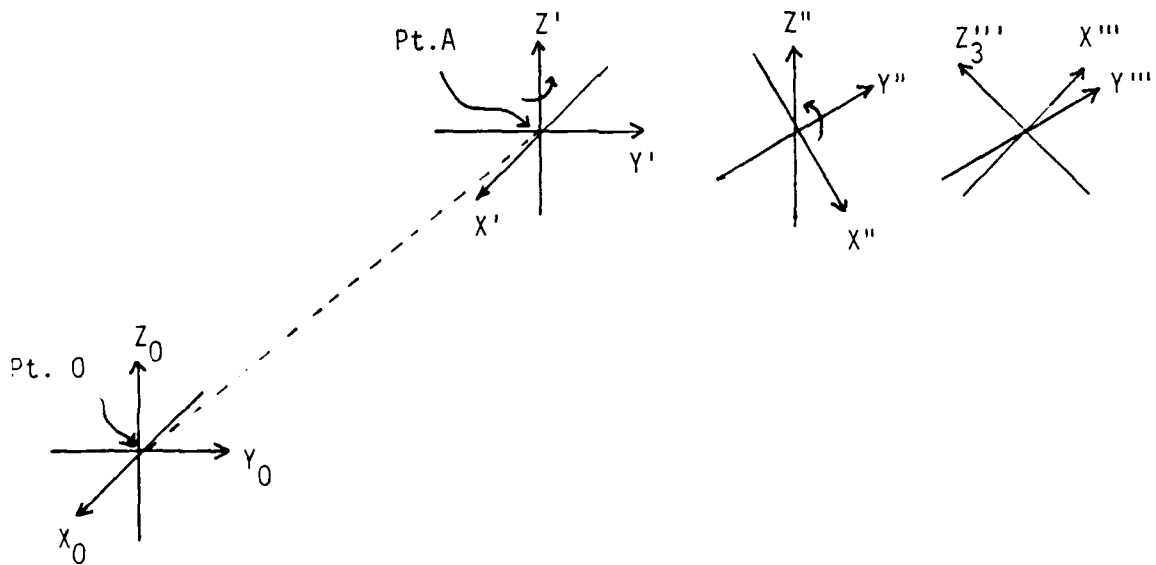


FIGURE 2.5 Coordinate Frames for the Wrist

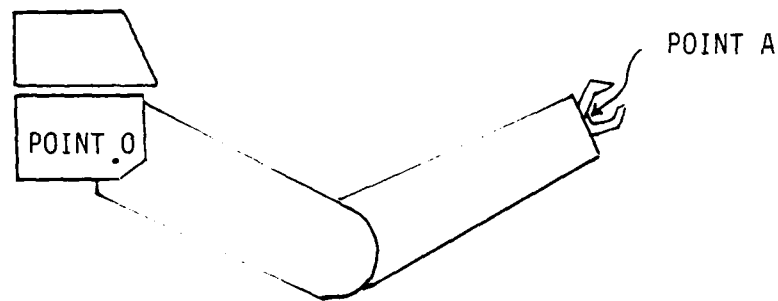


FIGURE 2.3 Point A on the Right Arm

arrow in each frame in Figure 2.4 signifies the 0 radian point.) The second rotation transformation is about the X_1 axis. The shoulder pitch or θ_2 can rotate from $\theta_2 = 1.31$ radians to 2.62 radians. The coordinate frame is then translated along the Y_2 axis by the length of the first link ($L_1 = .41$ meters). The elbow rotation, θ_3 , is the next transformation. It rotates about X_3 from $\theta_{3_{MIN}} = .52$ radians to $\theta_{3_{MAX}} = 3.75$ radians. The last transformation is a translation along the Y_4 axis the length of the second link ($L_2 = .36$ meters). This is point A.

The generation of a straight line path for point A is the main concern. However, the orientation angles (wrist yaw and wrist roll, θ_4 and θ_5) can also be changed while the arm is moving. The reference frame of the wrist is fixed to point A and shown in Figure 2.5. The first transformation is wrist yaw, θ_4 , which rotates about the Z' axis and varies from $\theta_{4_{MIN}} = 0$ radians to $\theta_{4_{MAX}} = 3.14$ radians. The next trans-

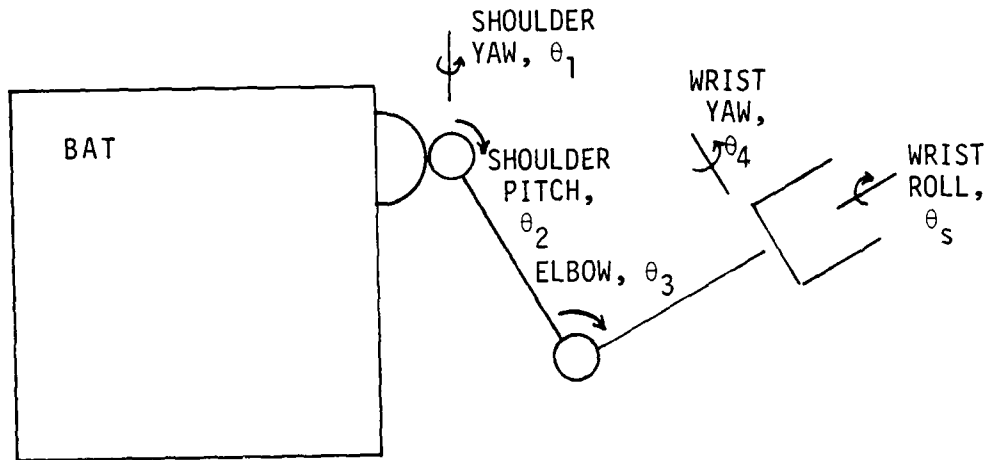


FIGURE 2.2 JOINT ANGLES

2.3 Problem Set-Up

The position of the end effector is the main concern. This is the point that must follow a straight line path (Point, A, Figure 2.3). This gives the arm three degrees of freedom (shoulder yaw, shoulder pitch and elbow). The restriction of point A to follow a straight line path from the start point to the finish point is a two-link problem. The reference frame (X_0 , Y_0 , Z_0) is positioned at the base of the first link (Point 0 in Figure 2.3). The first rotation transformation is about the Z_0 axis (Figure 2.4). This is shoulder yaw or θ_1 . It can rotate from $\theta_{1_MIN} = 0.0$ radians to $\theta_{1_MAX} = 2.44$ radians. (Note: the base of the

From this the central controller computes the initial position of the end effector. The procedure for converting the voltage inputs to joint angles will be covered in the next chapter. The final position is predetermined in advance and stored in the central controller. The problem is to find a trajectory for the end effector of the arm, compute the joint angles, and send them to the control system. The procedure for sending the joint angles to the control system will also be covered in the next chapter. Additionally, the position of the end effector is restricted to following a straight line path. This restriction is added to make it easier to concatenate paths (path concatenation is developed in Chapter 4).

The problem is now in the following format.

ASSUMPTIONS:

The reference frame is fixed to the BAT and stationary.

GIVEN:

- 1) Joint angles for the original arm configuration.
(Shoulder Yaw (θ_1), Shoulder Pitch (θ_2), Elbow (θ_3),
Wrist Yaw (θ_4) and Wrist Roll (θ_5), Figure 2.2).
- 2) The final position of the arm.
- 3) The physical dimensions of the arm.

FIND:

- 1) A straight line path for the position of the end effector from the start point to the finish point.
- 2) The joint angles for each position on the path.

determine what configuration the arm is in, and start generating inputs to the control system for a specific task. These inputs would mimic the inputs that would have come from the master arm if the master were controlling the task. An important task in an assembly operation is for the arm to move from a point (x,y,z) (in a three-dimensional space with a reference frame stationary and fixed to the BAT) to another point (x,y,z) . This thesis develops a method for the central controller to control the right arm moving from any point to a specified point. Using this method, it develops a way to concatenate specified points and move the arm through these points. This will enable the BAT to learn many trajectories which can be combined to perform assembly tasks. This will be the basis for the extensions to other types of tasks.

2.2 Overview of the Procedure

In this chapter, the task is to develop a method in which the arm moves to a specified location; for example, the operator has just grasped a beam with the right arm, and would like to move to a location next to the left arm. The central controller must:

- 1) know the starting configuration of the arm;
- 2) know the final configuration of the arm;
- 3) compute a trajectory;
- 4) transmit the appropriate command to the joint/actuator control system.

The operator starts this procedure by entering a keyboard option on the control panel at the ICS. This would let the central controller know that the supervisory control option has been selected. The original configuration is then read from the inputs of each joint of the master arm.

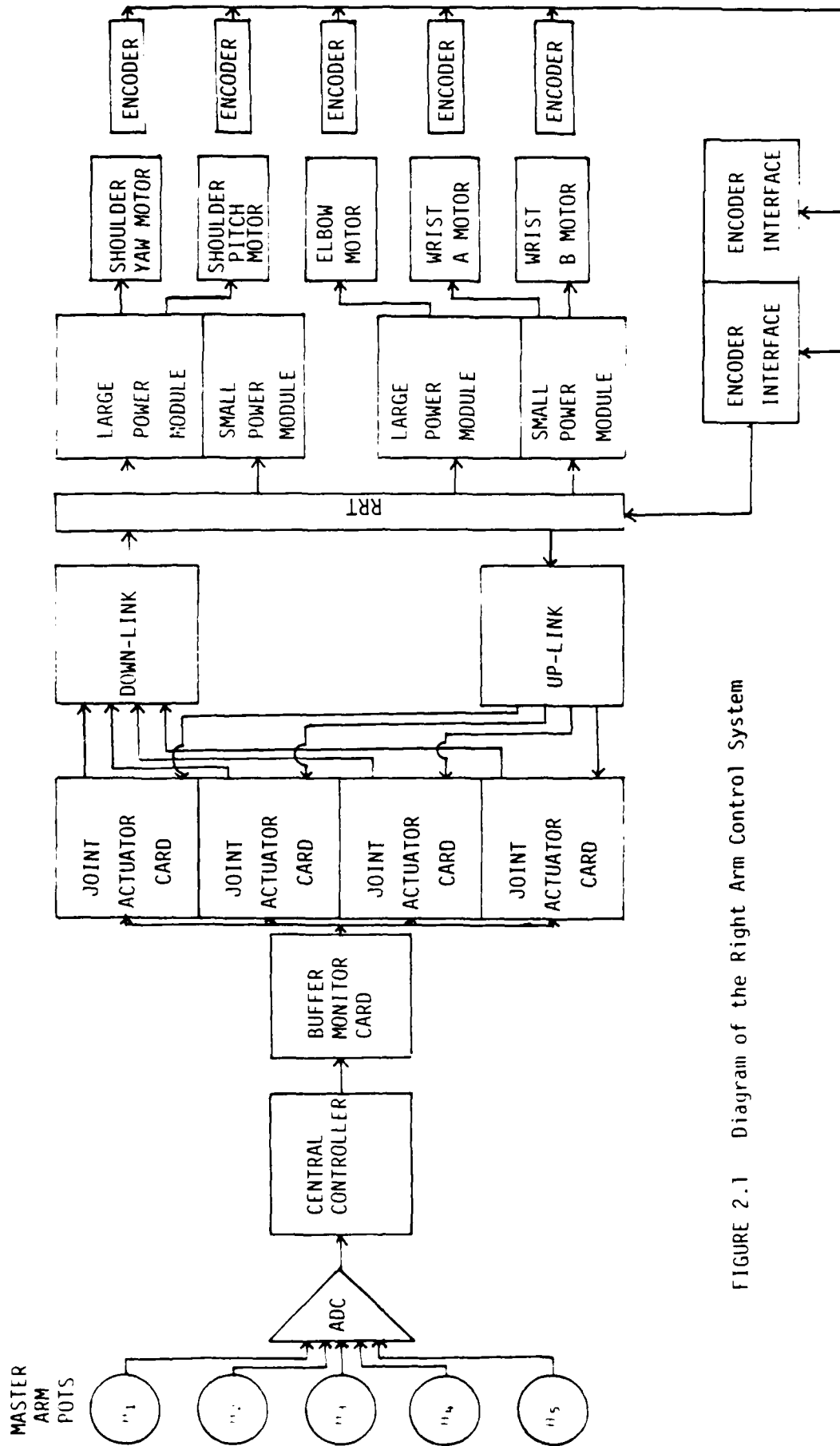


FIGURE 2.1 Diagram of the Right Arm Control System

CHAPTER 2

SUPERVISORY CONTROL OF THE RIGHT ARM

2.1 Objective

Using the two joysticks to control the BAT's position and the master arm to control the BAT's right arm, it is easy to imagine times where controlling the operation of the BAT can become a very hard task with only two hands. The operator might also need to enter an option in the BAT's software. The control system of the right arm can be improved, so that the master arm is not needed to perform most tasks. There are some cases in an assembly operation where the master-slave relationship would be easier to use. (For example, if two beams do not fit just right and have to be jiggled to make the connection.) However, in most cases it would be much easier to simply enter a command and let the central controller move the right arm to a preprogrammed position.

Figure 2.1 depicts the entire control system of the BAT's right arm. The position of the right arm is fed back only to the joint/actuator cards and not to the central controller. Therefore, the central controller must receive the positions of the right arm from the master arm at the ICS. The assumption is made that the master arm's joint angles are equal to the right arm's joint angles, or at least the relationship is a known linear one (recall the master arm is kinematically similar to the right arm). In the supervisory mode, the commands from the master arm are halted at the central controller. The central controller can then

with a specified start and finish point, the BAT system will then be taught to not only remember the trajectory, but also to play it back at the operator's command. This will give the BAT system the capability of performing any task over and over again without the use of the master arm. There are many tasks that are repetitive during an assembly operation. It would not be an efficient use of the BAT system if the operator were to perform these tasks again and again using the master arm at the ICS. By upgrading the control system, the BAT's capabilities will be greatly enhanced. Freeing the operator's hands will further increase the man-machine productivity.

degrees of freedom. It is also equipped with two cameras providing the operator with two different perspectives of the work area.

The BAT is controlled by the ICS. The ICS has two three degree-of-freedom joysticks to control the BAT's six maneuvering degrees of freedom. There is a master arm which is kinematically similar to the BAT's dexterous right arm. The operator uses this master arm to control the BAT's right arm in a master-slave control relationship. On the ICS there is also a helmet gimbal system used to control the tilt and pan unit of the BAT's main camera. The BAT and the ICS are described in further detail in Appendix A.

To perform an assembly task with the system, the operator uses the two joysticks to position the BAT at the work area. The operator then straps his/her right arm to the ICS's master arm and positions it to where he/she would like the BAT's right arm to be. The left arm is used to hold the workpiece in place. This can be quite challenging especially while the operator is trying to control the BAT's position with the joystick controls. A method of freeing the operator from the master-slave control relationship would help the man-machine to be more productive.

In this thesis, the operator is released from certain tasks by implementing supervisory control of the right arm. Two different trajectories will be used to demonstrate the control. The first trajectory will have an unknown starting point and a specified finishing point. By selecting a keyboard option, the operator will be able to move the BAT's right arm to this specified finish point. By defining a second trajectory

a more efficient and effective system. This thesis will continue along this trend to upgrade the interface between man and machine.

The teleoperator system is made up of two major structures:

- 1) the Beam Assembly Teleoperator (BAT)
- 2) the Integrated Control Station (ICS)

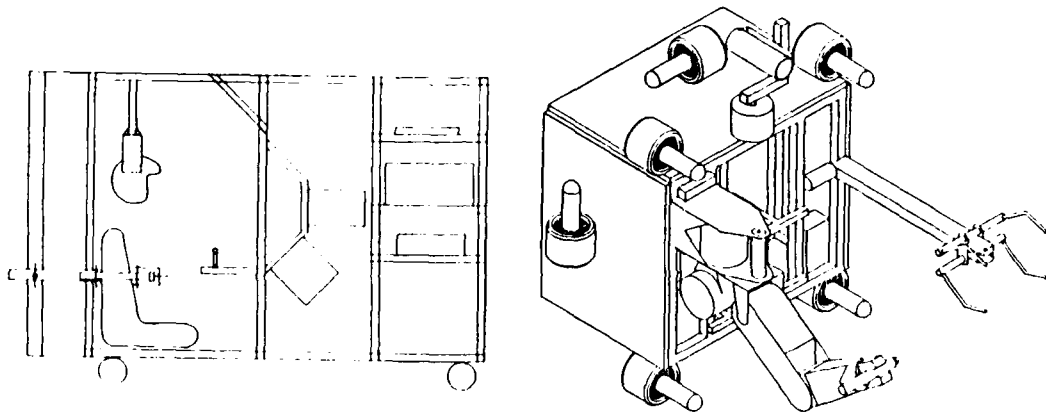


FIGURE 1.1 Sketch of the ICS and the BAT

The BAT has a fixed left arm and a dexterous right arm. The left arm has a gripper end effector used for grasping a beam, and a motor-driven roller used for moving the beam laterally. The right arm is a five degree-of-freedom manipulator with a two-jaw grip end effector. The BAT frame has eight troling motors which are used to give the unit six

CHAPTER 1

INTRODUCTION

One of the most important parameters affecting the industrialization of space is the capability to assemble structures in orbit. Much larger structures can be built in space than on earth. Since there is a limit to payload capacity, the need to assemble structures in space is imperative. Experimentation with assembly tasks on earth will help in the development of the most efficient assembly method, which will increase productivity on an actual mission. Tests have already been conducted, in a simulated environment, evaluating the ability of humans to assemble objects in space. The next step is to see how a machine can help man in accomplishing these tasks. The Space Systems Laboratory at M.I.T. has already conducted tests involving assembly by humans in a simulated environment. Neutral buoyancy was used as the simulation medium. A teleoperated system was also developed which will simulate the man-machine interface in space. This device is called the Beam Assembly Teleoperator (BAT). The BAT operates in the neutral buoyancy environment.

The Space Systems Laboratory began development of the BAT in mid 1981, and has been using this device in experiments with man-machine assembly tasks. The software for master-slave control of the BAT manipulator was previously developed in the Space Systems Lab. Both the hardware and the software are being continuously improved to provide

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
A.12	Joint/Actuator Control System	127
A.13	Propulsion Control System	129
A.14	Communication Link	130
A.15	Main() Function	132
A.16	Initializebat() Function	134
A.17	Calibratebat() Function	135
A.18	Screen() Function	137a
A.19a	Runbat() Function	137b
A.19b	Runbat() Function	138
A.20	Shutdownbat Function	146

yaw (θ_1) transforms into ψ identically. The transformation of shoulder pitch and elbow (θ_2 and θ_3) to R, Z is just a planar problem. The plane is defined by the location of A . For a particular value of θ the arm can only be in these positions.

- i) Elbow angle (θ_3) $< \pi$ (Elbow down position)
- ii) Elbow angle (θ_3) $> \pi$ (Elbow up position)
- iii) Elbow angle (θ_3) $= \pi$ (Fully extended)

Position i) Elbow angle (θ_3) $< \pi$ (Figure 2.7)

Note: There can never be a negative R so the arm is only in Quadrant I and IV of the (R, Z) plane.

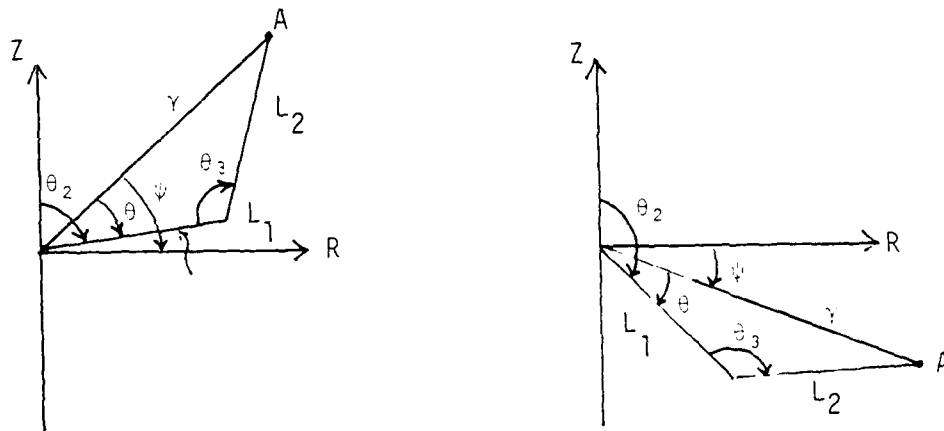


FIGURE 2.7 Joint Angle to Cylindrical Coordinates - Position i)

To find Z,R for either of the two diagrams in Figure 2.7 the following procedure is used:

- 1) find γ (the distance from the origin to point A) using the law of cosines,

$$\gamma = \sqrt{L_1^2 + L_2^2 - 2 \times L_1 \times L_2 \times \cos \theta_3}$$

- 2) find ϕ (the angle of the triangle formed by L_1 and L_2 at the origin) using the law of sines,

$$\phi = \sin^{-1} \left(\frac{L_2 \sin \theta_3}{\gamma} \right)$$

(NOTE: ϕ can only be between 0 and $\pi/2$ radians, therefore, there is no ambiguity in the arcsin.)

- 3) find ψ (the angle between γ and the R axis)

$$\gamma = \frac{\pi}{2} - (\theta_2 - \phi)$$

- 4) find Z,R using polar coordinate transformations

$$Z = \gamma \sin \psi$$

$$R = \sqrt{\gamma^2 - Z^2}$$

Position ii) Elbow angle (θ_3) $> \pi$ (Figure 2.8).

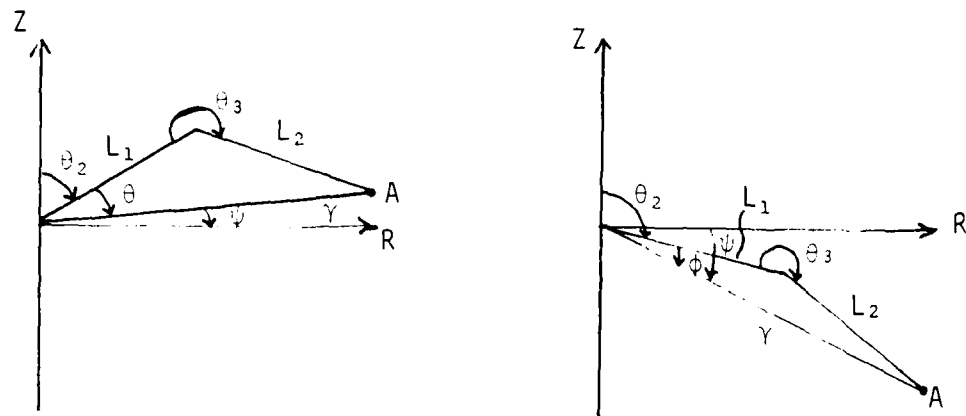


FIGURE 2.8 Joint Angle to Cylindrical Coordinates - Position ii

This configuration of the master arm is not possible given the current design of both the master arm of ICS and the dexterous arm of BAT. It is included in this thesis to ensure that this method can handle all types of inputs to the control system. (The algorithm for finding R and Z is similar to Position i).

- 1) Find γ (the distance from the origin to point A)
using the law of cosines

$$\gamma = \sqrt{L_1^2 + L_2^2 - 2 \times L_1 \times L_2 \times \cos(2\pi - \theta_3)}$$

(NOTE: The angle of the triangle is now $2\pi - \theta_3$ instead of θ_3 .)

- 2) find ϕ (the angle of the triangle formed by γ , L_2
at the origin) using the law of sines

$$\phi = \sin^{-1} \left(\frac{L_2 \sin(2\pi - \theta_3)}{\gamma} \right)$$

(NOTE: can only be between 0 and $\pi/2$.)

- 3) find ψ (the angle between γ and the R axis)

$$\psi = \frac{\pi}{2} - (\theta_2 - \phi)$$

- 4) find Z, R

$$Z = \gamma \sin \psi$$

$$R = \sqrt{\gamma^2 - Z^2}$$

Position iii) Elbow angle (θ_3) = π (Figure 2.9)

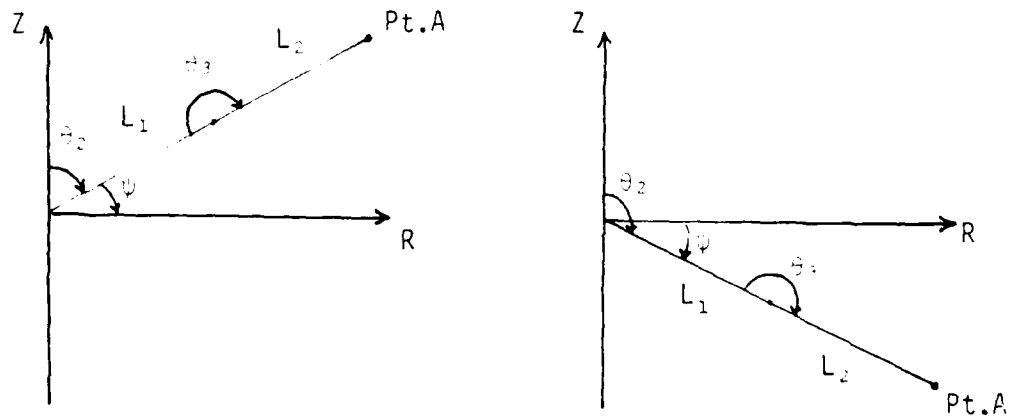


FIGURE 2.9 Joint Angles to Cylindrical Coordinates - Position iii

- 1) find γ (the distance from the origin to point A)

$$\gamma = L_1 + L_2$$

- 2) find ψ (the angle between γ and the R axis)

$$\psi = \frac{\pi}{2} - \theta_2$$

- 3) find Z, R

$$Z = \gamma \sin \psi$$

$$R = \gamma \cos \psi$$

Upon receiving the joint angles (shoulder yaw, θ_1 , shoulder pitch, θ_2 , and elbow, θ_3) the controller checks θ_3 and finds θ, R, Z of the initial position of point A from one of these three algorithms.

In this chapter the final position of point A is known and has been stored in the controller. The final position is stored in cylindrical coordinates. The final orientation angles θ_4, θ_5 (wrist yaw and wrist roll) are also stored and are in radians. The controller knows it must move point A from θ, R, Z (initial) to θ, R, Z (final) in a straight line.

2.5 Straight Line Path Generation

If a path is a straight line in X_0, Y_0, Z_0 space, it can be projected down to a straight line in the X, Y plane (Fig. 2.10). Due to the maximum and minimum values of shoulder yaw (θ_1) only the first and second quadrants of the (X_0, Y_0) plane are involved. Having θ, R initial and θ, R final, measured as shown in Figure 2.11, the values for X, Y initial and X, Y final are easily computed.

<u>Initial</u>	<u>Final</u>
$Y_i = R_i \times \sin \theta_i$	$Y_f = R_f \times \sin \theta_f$
$X_i = R_i \times \cos \theta_i$	$X_f = R_f \times \sin \theta_f$

From here the equation for a straight line is then computed in terms of X and Y .

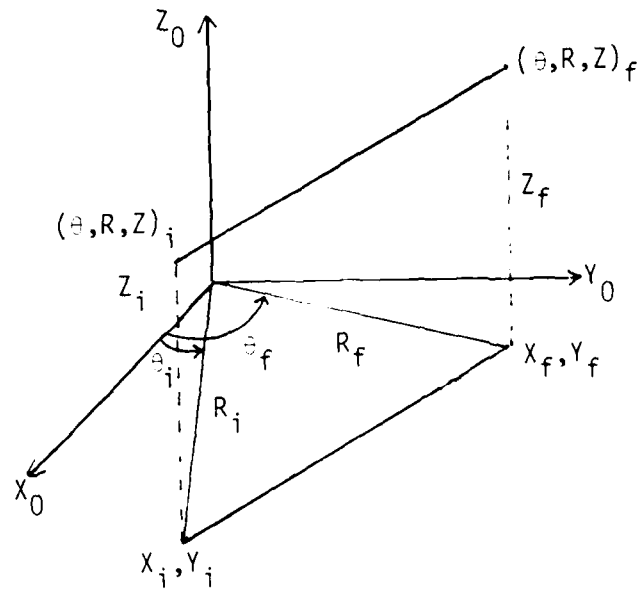


FIGURE 2.10 Straight Line Path in Reference Frame (X_0, Y_0, Z_0)

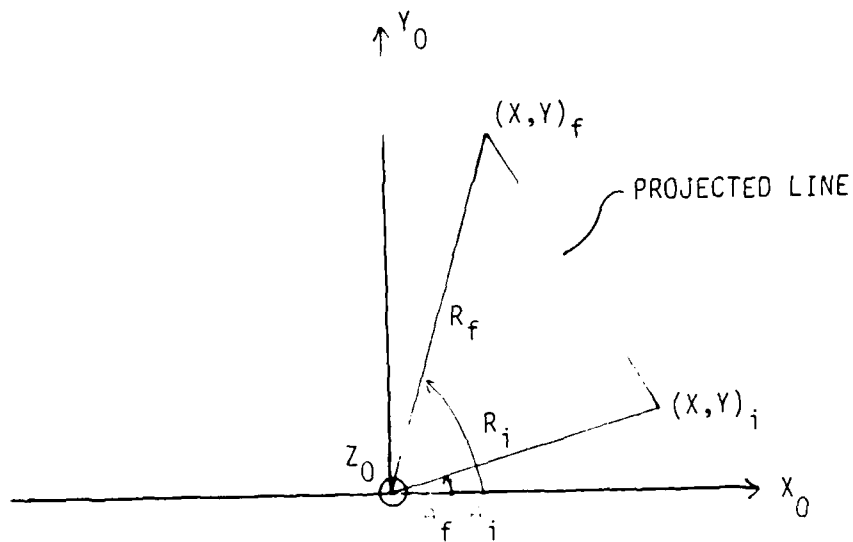


FIGURE 2.11 Projected Line in X_0Y_0 Plane

$$Y = m(X - X_i) + Y_i \quad (2.1)$$

where

$$m = \frac{Y_f - Y_i}{X_f - X_i} \quad .$$

[The case when $\Delta X = 0$ sets the flag.] The value for θ changes every time step. The method for changing θ (also shoulder yaw, θ_1) will be discussed in Section 2.7. Assume for now that the value of θ at each time step is given. Then R and Z are computed using the X and Y computed from the straight line restriction (2.1). (See Figure 2.12.)

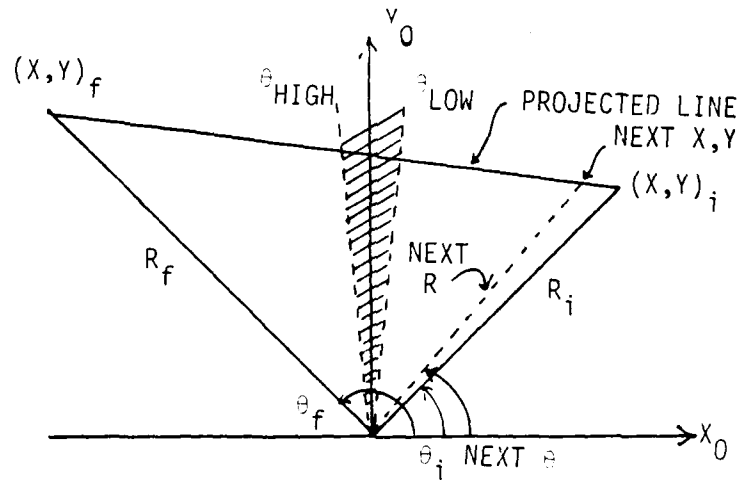


FIGURE 2.12 Travelling Across the Projected Line

To review what has been done until now, first the joint angles $(\theta_1, \theta_2, \theta_3)$ were obtained from the arm. The joint angles were then converted to cylindrical coordinates of point A. With the initial and final cylindrical coordinates of point A, a straight line path in the X_0, Y_0, Z_0 plane was found for point A. As θ (which is also the shoulder yaw angle, θ_1) sweeps across the straight line, the value of R and Z must be found for each θ . This will give the cylindrical coordinates of point A on the straight line. Knowing the value of θ , R and Z of point A, this will be used to convert back to joint angles (this part will be covered in Section 2.6). To find the values of R and Z for each θ , four special cases may be examined.

Case 1) $\Delta\theta, \Delta Z = 0$

Here, θ and Z retain the value of θ_i and Z_i . Depending on whether R_i is less than or greater than R_f , the appropriate driving function is used to determine R at each time step (this driving function is the same one used to determine θ when $\Delta\theta \neq 0$ and will be covered in Section 2.7).

Case 2) $\Delta\theta, \Delta R = 0$

This is done exactly as above except that R is set equal to R_i and Z is used in the driving function.

Case 3) $\Delta\theta = 0$

For this condition, R is used in the driving function. X and Y are found using simple conversions.

$$X = R \times \cos \theta$$

$$Y = R \times \sin \theta .$$

Dependent upon the value of ΔX and ΔY the appropriate side of the equation for a straight line (shown below) is used to compute Z .

$$\frac{X - X_i}{X_f - X_i} = \frac{Z - Z_i}{Z_f - Z_i} = \frac{Y - Y_i}{Y_f - Y_i} .$$

Case 4) If $\Delta R, \Delta Z = 0$

or $\Delta R = 0$

or $\Delta Z = 0$

or $\Delta\theta, \Delta R, \Delta Z \neq 0$

the following procedure is used.

The value of each θ , given by the driving function, is used to compute the next X and Y along the straight line. From these two values, R is computed as shown below.

General case:

using

$$X = \frac{Y}{\tan \theta}$$

substitute this into the equation for a straight line (2.1)

$$Y = m \left(\frac{Y}{\tan \theta} - X_i \right) + Y_i \quad .$$

Then, solving for Y

$$Y = \frac{(Y_i - m X_i)}{\left(1 - \frac{m}{\tan \theta}\right)} \quad .$$

Now with Y and θ , find X

$$X = \frac{Y}{\tan \theta} \quad .$$

R is then found by

$$R = \sqrt{X^2 + Y^2} \quad .$$

Flagged cases:

θ is checked as it changes, so that when the next θ reaches a value of θ_{low} , (Figure 2.12, in this case $\theta_{low} = 1.53$ radians) θ retains this value until it passes $\theta = \pi/2$. At $\theta = \pi/2$, θ is then set to θ_{high} which is 1.60 radians. This is to account for the singularity at $\theta = \pi/2$. There are four more cases which are flagged.

Case 1) If $\Delta x = 0$ (Figure 2.13)

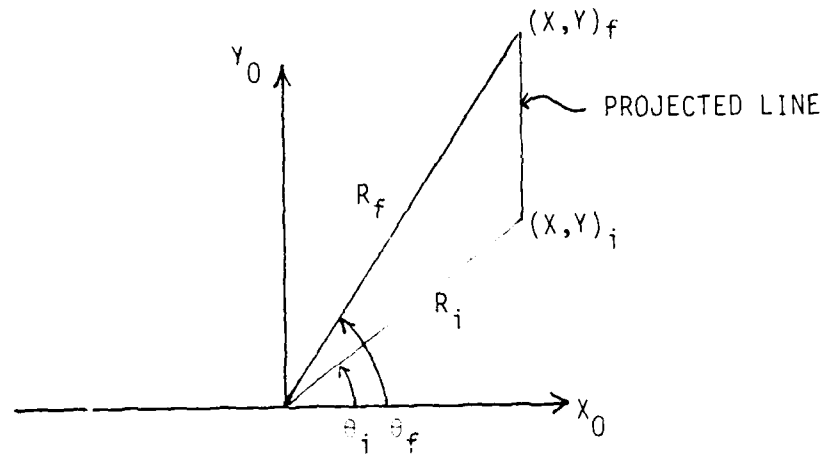


FIGURE 2.13 Slope of the Projected Line Equal to 1

Then,

$$Y = X_i \tan \theta$$

$$X = X_i$$

Case 2) If $m = 0$

Then

$$X = \frac{Y_i}{\tan \theta}$$

$$Y = Y_i$$

Case 3) If $\theta = 0$

Then

$$Y = 0 \quad X = X_i - \frac{Y_i}{m} \quad (\text{except when } m = 0)$$

Case 4) If $\theta = \frac{\pi}{2}$ (possibly as a starting point)

Then

$$X = 0$$

$$Y = Y_i - m X_i \quad .$$

Once X and Y are known, using the general or flagged cases, R is then found using

$$R = \sqrt{X^2 + Y^2} \quad .$$

To find Z , the equation for a straight line in three-dimensional space is

$$\frac{X - X_i}{X_f - X_i} = \frac{Z - Z_i}{Z_f - Z_i} = \frac{Y - Y_i}{Y_f - Y_i} \quad .$$

The values of Z_i and Z_f are known and depending on whether ΔX or ΔY is zero the right or left side of the equation is used.

$$Z = \left(\frac{X - X_i}{X_f - X_i} \right) (Z_f - Z_i) + Z_i \quad Z = \left(\frac{Y - Y_i}{Y_f - Y_i} \right) (Z_f - Z_i) + Z_i$$

2.6 Cylindrical Coordinates to Joint Angles

At this point, the values of θ, R, Z which keep point A on a straight line path are known. Therefore for a particular value of θ ,

finding θ_2 (shoulder pitch) and θ_3 (elbow) is a planar problem. Recall that θ in cylindrical coordinates is equal to θ_1 (shoulder yaw). The algorithm to find the joint angles is simply the reverse of going from $(\theta_1, \theta_2, \theta_3)$ joint angles to (θ, R, Z) cylindrical coordinates. Only three positions can occur here also.

Position i) Elbow angle $(\theta_3) < \pi$ (Figure 2.14)

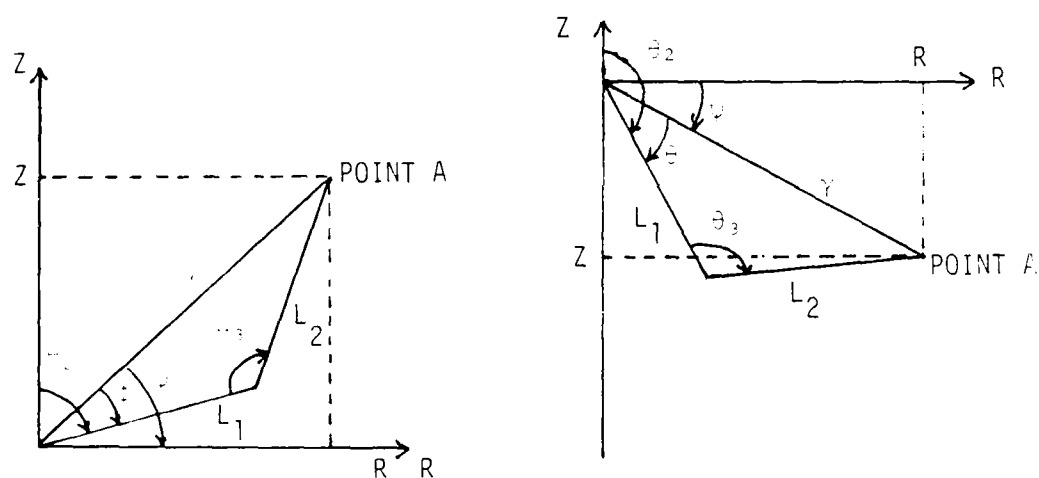


FIGURE 2.14 Cylindrical Coordinates to Joint Angles - Case i

1) find r

$$r = \sqrt{R^2 + Z^2}$$

If the case arises where $\Delta\theta, \Delta R = 0$ (case 2, Section 2.5), T is again computed using the maximum $\Delta\theta = \pi$. The driving function will now be $Z(t)$.

$$Z_f = Z_i$$

$$Z(t) = Z_i + (\Delta Z(\frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

$$Z_f = Z_i$$

$$Z(t) = Z_f - (\Delta Z(1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

2.8 Determining the Orientation Angles

The values of the orientation angles θ_4, θ_5 (wrist yaw and wrist roll) do not affect the straight line path of point A. Therefore, wrist yaw, θ_4 , and wrist roll, θ_5 , can be changed slowly as the arm moves point A along its straight line path. Recall from the end of Section 2.3, θ_{4i} and θ_{5i} (initial) are obtained by transforming the values of ϕ_4 (wrist motor A) and ϕ_5 (wrist motor B). The values of the final orientation angles, θ_{4f} and θ_{5f} , are stored along with the final position of point A. In order to have all of the joints of the arm finish at the same time, the time it takes to go from θ_{4i}, θ_{5i} (initial) to θ_{4f}, θ_{5f} (final) will be T (the total time to traverse the straight line path, Section 2.7). To account for gradual acceleration and braking of the two

increment, will be the sampling time of the arm's control system ($\Delta t \approx .05$ seconds). It is obvious, by looking at Figure 2.19, that gradual acceleration and gradual braking will be applied to the shoulder yaw. At each time increment the values for t, T and $\Delta\theta$, are used in the driving function to compute the value of $\theta_1(t)$. This value of θ_1 is then used to determine R and Z of point A on a straight line path as shown in Section 2.5.

For case 3, when $\theta_{1i} = \theta_{1f}$, the driving function, $\theta_1(t)$ cannot be used. In this case T is computed using the maximum change, $\Delta\theta_1 = \pi$. The driving function will then be determined by the change in radius, ΔR . The same type of driving function is used for $R(t)$ as was used for $\theta_1(t)$ in case 1 or 2. This will also provide gradual acceleration and braking of the arm.

$$R_f > R_i$$

$$R(t) = R_i + (\Delta R (\frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

$$R_f < R_i$$

$$R(t) = R_f - (\Delta R (1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T})))$$

The values of X, Y and Z are then computed as in case 3, Section 2.5 (also for case 1).

small. Too small a time might cause the controller to generate a command that could saturate the actuator. Therefore, during experimentation with this method on the BAT, the value of FS is approximately 3.5. It starts out this high for equipment safety and can be decreased if the arm moves too slowly. Once the value of $\Delta\theta$ (the change in shoulder yaw) is computed from the initial and final position of point A, it is used to compute T , the total time to traverse the straight line path.

The value of T along with the value of $\Delta\theta$ (the shoulder yaw change) is used in the driving function, $\theta(t)$ (Figure 2.17). By letting t start from 0 and slowly increasing t until $t = T$, the driving function, $\theta(t)$, will be sampled [1]. This is shown in Figure 2.19 for the case when θ_{1f} is greater than θ_{1i} . The value of Δt , the time

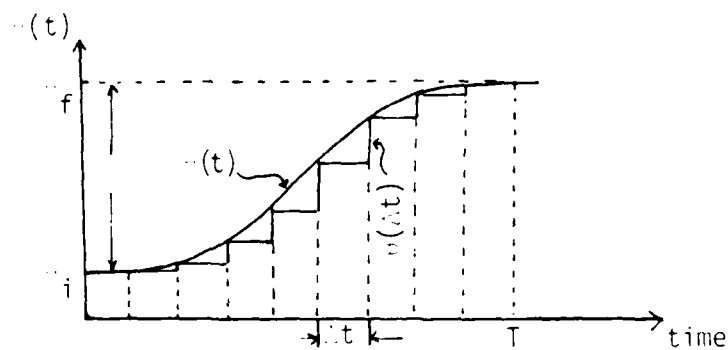


FIGURE 2.19 Sampled Driving Function

Integrate over X to find the total drag force. To find the Drag Moment,

$$\text{Drag Moment} = \text{Drag Force} \times \text{Moment Arm}$$

$$\text{Drag Moment Arm} = D_{\text{ARM}}^Y \quad Y : 0, L_{\text{ARM}}$$

$$\text{Drag Moment Beam} = D_{\text{BEAM}}^Y \quad Y : L_{\text{ARM}}, L_{\text{ARM}} + \text{Diameter of Beam}$$

Integrating over Y gives the total drag moment on the arm and beam for a given velocity, V . The two integrations give a constant (called Drag) multiplied by the $\dot{\theta}^2$ term

$$\text{Drag Moment} = \text{Drag} \dot{\theta}^2$$

The value for maximum torque (τ) was computed using the maximum torque for the shoulder yaw motor and the gear train dimensions between the shoulder and the arm. To find the total time, the maximum acceleration is used along with the velocity at that time. Recall that the maximum acceleration occurs at $t = \frac{T}{4}$. The value for t is substituted into the equations for acceleration and velocity (Figure 2.17). These values of acceleration and velocity are used with the constants (J_T , $\Delta\theta$, Drag) to find the equation for T (Eq. (2.2)). FS is a factor of safety which is used to ensure that the time taken to traverse the straight line path is not too

Moment of Inertia:

$$J_T = J_{ARM} + J_{BEAM}$$

$$J_{ARM} = J_{GA} + m_{ARM} \left(\frac{L_{ARM}}{2} \right)^2$$

$$J_{GA} = \frac{1}{12} m_{ARM} (3 R_{ARM}^2 + L_{ARM}^2)$$

$$J_{BEAM} = J_{GB} + m_{BEAM} \left(L_{ARM} + \frac{R_{BEAM}}{2} \right)^2$$

$$J_{GB} = \frac{1}{2} m_{BEAM} R_{BEAM}^2$$

Drag Moment:

$$\text{Total Drag Force} = D_{ARM} + D_{BEAM}$$

where

$$D = \frac{\rho C_D A V^2}{2}$$

ARM

$$\text{Velocity} = \dot{\theta}(x)$$

$$X : 0, L_{ARM}$$

$$A = (\text{Diameter})(X)$$

$$X : 0, L_{ARM}$$

BEAM

$$\text{Velocity} = \dot{\theta}(L_{ARM} + X)$$

$$X : 0, \text{Diameter of Beam}$$

$$A = (L_{BEAM})(X)$$

$$X : 0, \text{Diameter of Beam}$$

The values of J_T (total moment of inertia of the arm and beam) and drag moment exerted on the arm and beam by the medium are constants and computed using the simplified model shown in Figure 2.18. (The drag moment is a constant when using the value for velocity at $t = \frac{T}{4}$.

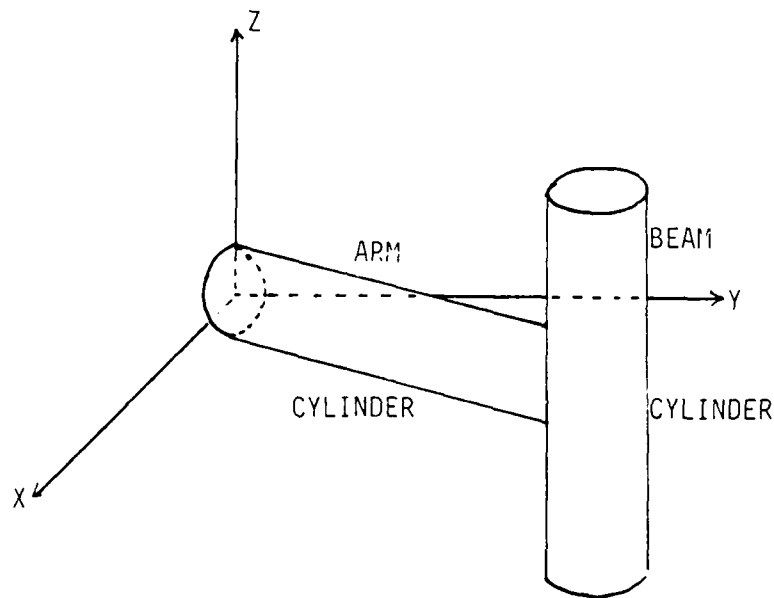


FIGURE 2.18 Simplified Model of Arm and Beam

By overestimating the dimensions of the arm and the beam, and assuming the arm and beam are in the position shown in Figure 2.18, the values of J_T and the drag moment are also overestimated. These two constants were easily computed using very straightforward equations.

the path is not of the utmost importance. The main concern is to make sure a command is not generated which will saturate the actuator. Therefore, as long as the values used in the calculation of T are over-estimated, the arm will move at a reasonable speed and the command will not saturate the actuator. To find T , the equation for the torque and drag force is used and solved for T .

$$\tau = J_T \ddot{\theta} + \text{Drag Moment}$$

where

τ = the maximum torque applied to the arm

J_T = the total moment of inertia of the arm grasping a beam

Drag Moment = the total drag moment on the beam and the arm.

at $t = \frac{T}{4}$

$$\ddot{\theta}(t) = \frac{2\pi|\Delta\theta|}{T^2}$$

$$\dot{\theta}(t)^2 = \frac{|\Delta\theta|^2}{T^2}$$

$$\tau = J_T \ddot{\theta}(t) + \dot{\theta}^2 \text{ Drag}$$

Substituting and solving for T

$$T = \text{F.S.} \sqrt{\frac{1}{\tau} [J_T 2\pi|\Delta\theta| + |\Delta\theta|^2 \text{ Drag}]} \quad (2.2)$$

where F.S. = a Factor of Safety.

Case 1) Shoulder yaw initial (θ_{1_i}) < shoulder yaw final (θ_{1_f}).

Case 2) Shoulder yaw initial (θ_{1_i}) > shoulder yaw final (θ_{1_f}).

The velocity and acceleration are simply the first and second derivatives. To account for the starting position not being at zero in Figures 2.17a and 2.17b, the two equations are adjusted as follows.

Shoulder yaw initial < shoulder yaw final

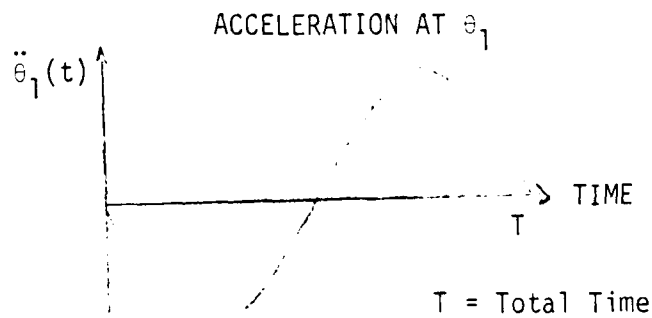
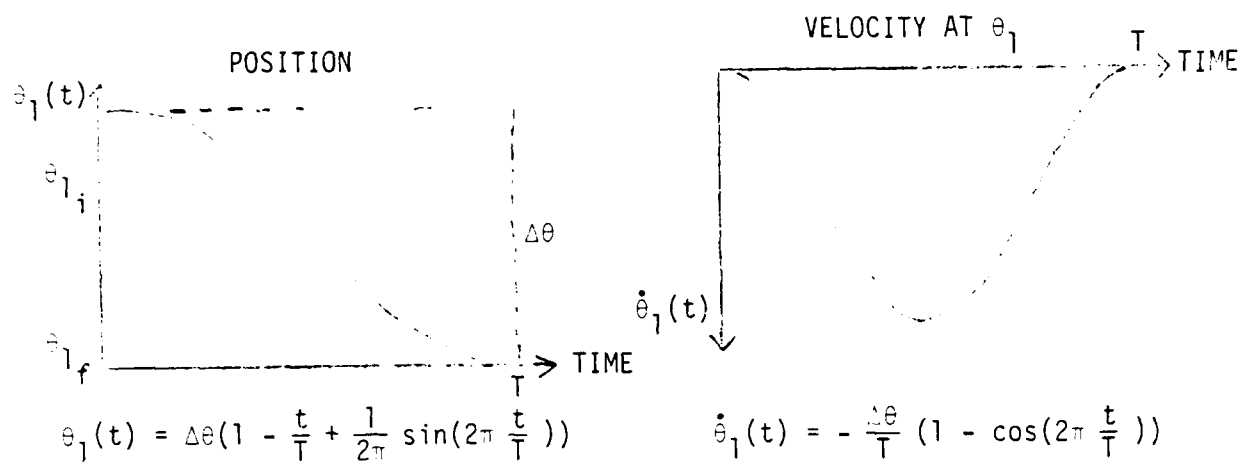
$$\theta_1(t) = \theta_{1_i} + (|\Delta\theta| \left(\frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

Shoulder yaw initial > shoulder yaw final

$$\theta_1(t) = \theta_{1_f} + (|\Delta\theta| \left(1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

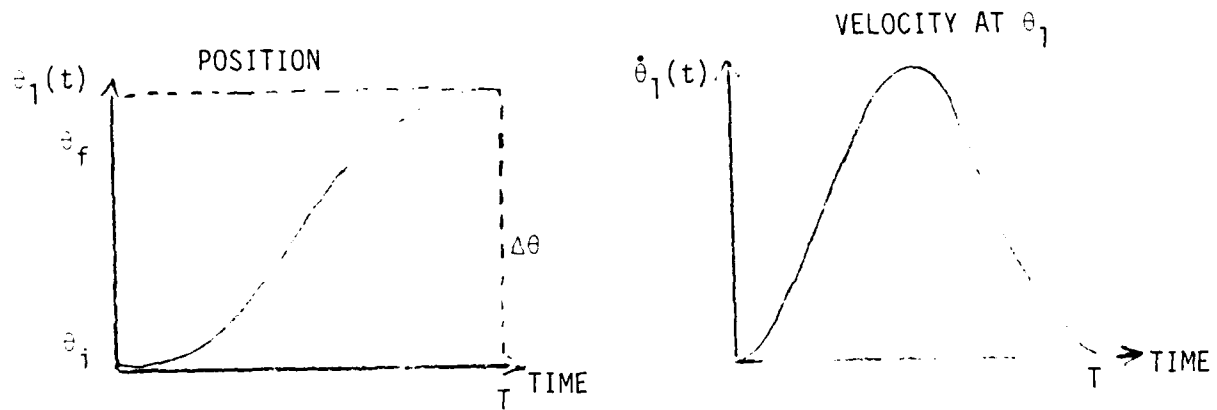
Case 3, when $\theta_{1_i} = \theta_{1_f}$, will be discussed at the end of this section.

By knowing the maximum acceleration the motor is capable of producing under the worst conditions, T (the total time it takes the arm to traverse the path) can be determined for a value of $\Delta\theta$. It is apparent that the maximum acceleration occurs at $t = \frac{T}{4}$ and $t = \frac{3T}{4}$. $t = \frac{T}{4}$ will be used to compute T , the total time. Putting $t = \frac{T}{4}$ into the equations for $\dot{\theta}_1(t)$ and $\ddot{\theta}_1(t)$, the maximum acceleration and velocity at that time are now functions of $\Delta\theta$ and T . In this thesis, the speed to complete



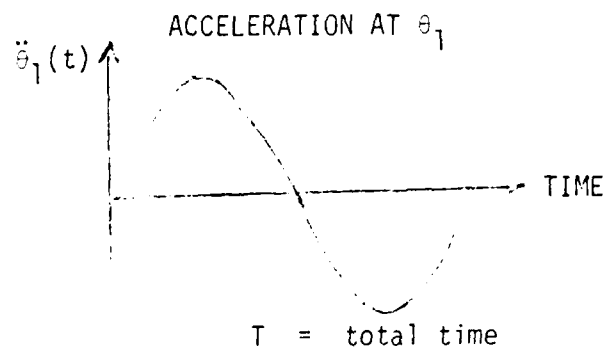
$$\ddot{\theta}_1(t) = -\frac{2\pi \Delta\theta}{T^2} \sin\left(2\pi \frac{t}{T}\right)$$

FIGURE 2.17b Driving Function (Initial Greater than Final)



$$\theta_1(t) = \Delta\theta \left(\frac{t}{T} - \frac{1}{2\pi} \sin \left(2\pi \frac{t}{T} \right) \right)$$

$$\dot{\theta}_1(t) = \frac{\Delta\theta}{T} (1 - \cos(2\pi \frac{t}{T}))$$



$$\ddot{\theta}_1(t) = \frac{2\pi \Delta\theta}{T^2} (\sin(2\pi \frac{t}{T}))$$

FIGURE 2.17a Driving Function (Initial Less than Final)

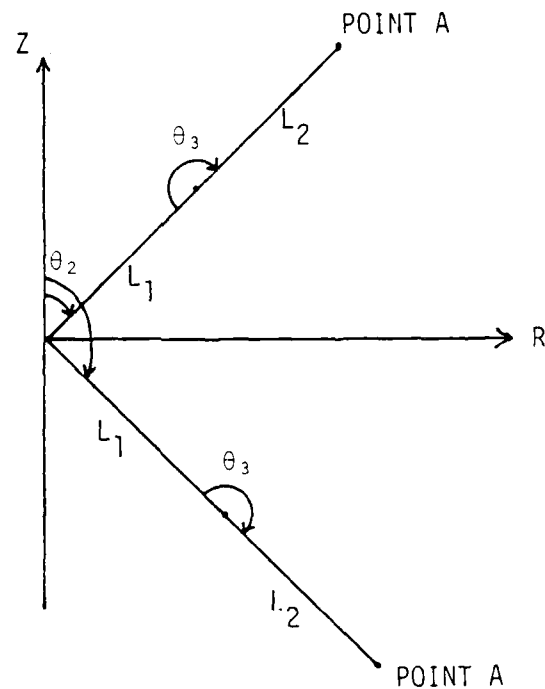


FIGURE 2.16 Cylindrical Coordinate to Joint Angles - Case iii

2.7 Determine the Total Time and the Value of Shoulder Yaw (θ_1) at Each Time Step

Knowing θ (initial) and θ (final) from the end of Section 2.4, gives the initial and final shoulder yaw angles. The motor at the reference base of the arm drives the shoulder yaw, θ_1 . The specifications of this motor and a simplified model of the arm will determine the total time it takes the arm to traverse the straight line path. With the initial and final value of θ_1 , a driving function, $\theta_1(t)$, is used to provide ample braking of the system (Figure 2.17a and 2.17b) [2]. The braking will be shown later when this function is sampled. Three cases can occur.

3) find θ_3 (elbow)

$$\theta_3 = 2\pi - \cos^{-1} \left(\frac{L_1^2 + L_2^2 - \gamma^2}{2 L_1 L_2} \right) .$$

Note: The angle in the triangle is only from 0 to π .

4) find ϕ

$$\phi = \sin^{-1} \left(\frac{L_2 \sin(2\pi - \theta_2)}{\gamma} \right)$$

ϕ is only from 0 to $\frac{\pi}{2}$.

5) find θ_2 (shoulder pitch)

$$\theta_2 = \frac{\pi}{2} - (\psi + \phi) .$$

Position iii) Elbow angle (θ_3) = π (Figure 2.16)

This case cannot happen in a straight line path unless the controller computes a γ greater than γ_{\max} due to an input error or if the manipulator starts out with $\theta_3 = \pi$. In either case, the algorithm switches to Position i). Through simple trigonometric routines, the values of $\theta_1, \theta_2, \theta_3$ (shoulder yaw, shoulder pitch and elbow) are found which will move the end effector along a straight line path.

Position ii) Elbow angle (θ_3) $> \pi$ (Figure 2.15)

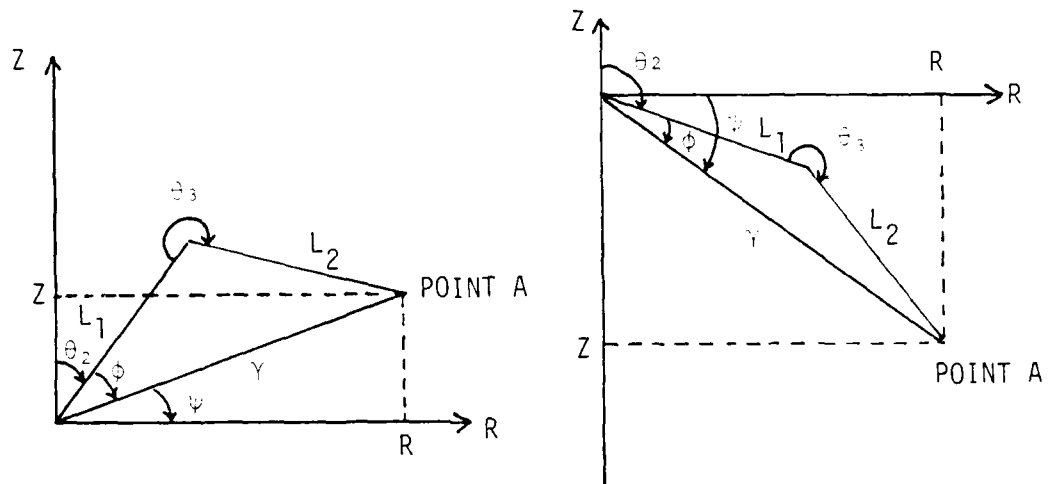


FIGURE 2.15 Cylindrical Coordinates to Joint Angles - Case ii

1) find γ

$$\gamma = \sqrt{Z^2 + R^2}$$

2) find ϕ

$$\phi = \tan^{-1} \left(\frac{Z}{R} \right)$$

ϕ is only from $-\pi/2$ to $\pi/2$.

γ is checked here to insure it does not exceed γ_{\max} . This is just in case the operator mistakingly enters in a point beyond the manipulator's reach. If $\gamma > \gamma_{\max}$, γ_{\max} is used and θ_3 (elbow) will be at π .

- 2) find ψ

$$\psi = \tan^{-1} \left(\frac{Z}{R} \right) .$$

Note: ψ can only go from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, therefore, the arctan function will have no ambiguity.

- 3) find θ_3 , using the law of cosines (elbow)

$$\theta_3 = \cos^{-1} \left(\frac{1}{2L_1L_2} (L_1^2 + L_2^2 + \gamma^2) \right) .$$

Note: The value of θ_3 can only go from 0 to π so the value of the arc-cosine function will have no ambiguity.

- 4) find ϕ , using the law of sines

$$\phi = \sin^{-1} \left(\frac{L_2 \sin \theta_3}{\gamma} \right) .$$

Note: The value of ϕ can only be between 0 and $\frac{\pi}{2}$.

- 4) find θ_2 (shoulder pitch)

$$\theta_2 = \frac{\pi}{2} - (\psi - \phi) .$$

wrist motors, the values of θ_4 and θ_5 at each time step are obtained using the same type of driving functions as $\theta_1(t)$. For example, if $\theta_{4f} > \theta_{4i}$ and $\theta_{5f} < \theta_{5i}$ then the values at each time step for θ_4 and θ_5 will be found using

$$\theta_4(t) = \theta_{4i} + (\Delta\theta_4 \left(\frac{t}{T} - \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

$$\theta_5(t) = \theta_{5f} - (\Delta\theta_5 \left(1 - \frac{t}{T} + \frac{1}{2\pi} \sin(2\pi \frac{t}{T}) \right))$$

where T = the total time found in Section 2.7

t = the value for time using the time increment, Δt .

The two values of orientation angles θ_4, θ_5 , are then converted to wrist motor (A and B) angles ψ_4, ψ_5 , using the transformations shown at the end of Section 2.3.

2.9 Simulation

An algorithm implementing this straight line path program was written in C and simulated on a VAX 11/782. The initial values of the joint angles and the final values of point A and orientation angles were entered from the keyboard. The program follows this outline:

- 1) Read the initial joint angles (shoulder yaw, θ_1 , shoulder pitch, θ_2 , elbow, θ_3 , wrist motor, ψ_4 , and wrist motor, ψ_5) from the keyboard (all values were entered in radians).

- 2) Convert joint angles $(\theta_1, \theta_2, \theta_3)_i$ to cylindrical coordinates $(\theta, R, Z)_i$ for the initial position.
- 3) Convert the wrist motor angles (ψ_4, ψ_5) to wrist orientation angles (wrist yaw, θ_4 , and wrist roll, θ_5) for the initial position.
- 4) Enter the final position of point A in cylindrical coordinates $(\theta, R, Z)_f$ and the final orientation angles $(\theta_{4f}, \theta_{5f})$. All values are entered in meters and radians.
- 5) Compute constants $(\Delta\theta, \Delta R, \Delta Z$, the initial and final position of point A in the X, Y, Z plane, the equation for the straight line between the two points).
- 6) Using $|\Delta\theta_1|$ (the change in shoulder yaw, which is also $\Delta\theta$ in cylindrical coordinates), compute T (the total time to traverse the straight line path).
- 7) Start a loop (from $t=0$ until $t=T$, incrementing t by Δt (where Δt used in simulation was .25 seconds instead of the sampling time ($\approx .05$ seconds), this was done for the sake of brevity during simulation).
 - a) Compute $\theta_1(t)$, the driving function, for the appropriate case as in Section 2.7.
 - b) Compute R and Z for point A at that particular θ (found in step 7a) where θ , R and Z are constrained by the straight line equation obtained in step 5.

- c) Compute the three joint angles (shoulder yaw, θ_1 , shoulder pitch, θ_2 , and elbow, θ_3) from the cylindrical coordinates (θ, R, Z) found in step 7b.
- d) Calculate the wrist yaw, θ_4 , and the wrist roll, θ_5 , using the appropriate driving functions, $\theta_4(t)$ and $\theta_5(t)$.
- e) Check the limits for all joint and orientation angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$).
- f) Convert wrist orientation angles (θ_4, θ_5) to wrist motor angles (ψ_4, ψ_5).
- g) Compute the velocity and acceleration of the shoulder yaw using the equations in Section 2.7.
- h) Plot the following graphs:
 - shoulder yaw versus time
 - shoulder yaw velocity versus time
 - shoulder yaw acceleration versus time
 - wrist yaw versus time
 - wrist roll versus time
 - the position of point A in X,Y,Z space
 - X versus Y
 - X versus Z
 - Y versus Z .

Many simulations were performed using this algorithm with excellent results. The plots show how gradual acceleration and braking were applied to shoulder yaw, θ_1 , wrist yaw, θ_4 , and wrist roll, θ_5 . The plots of point A in the X,Y,Z plane show how the path of point A is a straight line. Three of these simulation results follow.

Simulation 1) -Shoulder yaw initial, θ_{1i} , < shoulder yaw final, θ_{1f}
-the elbow is in the elbow down position
($\theta_3 < \pi$). (See Figure 2.20.)

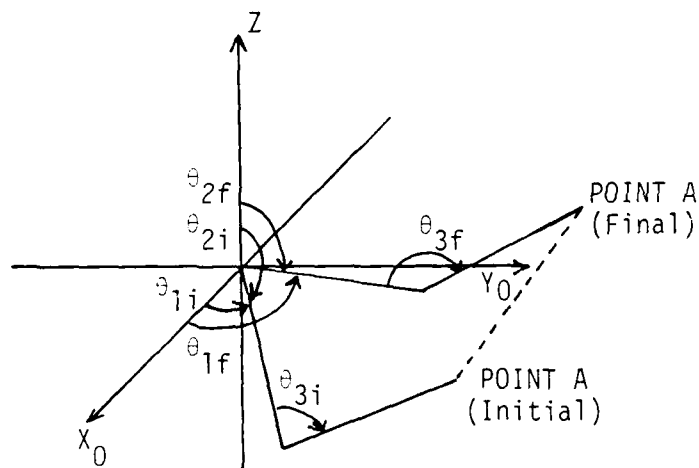


FIGURE 2.20 Arm Position Simulation I

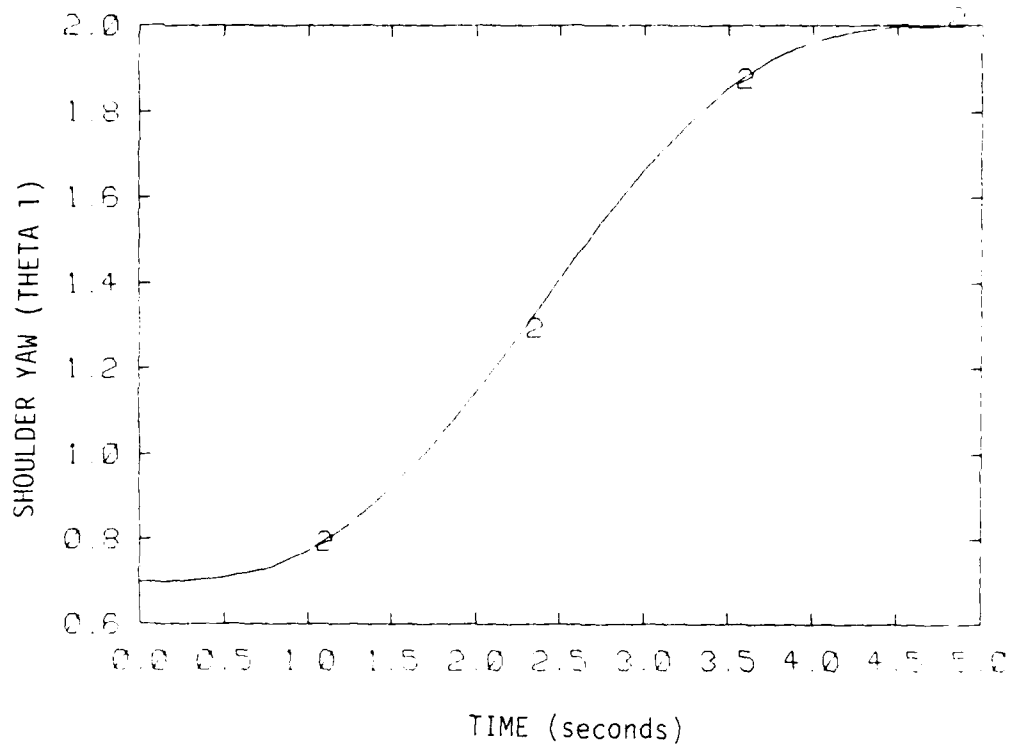


FIGURE 2.21 Shoulder Yaw Position - Simulation I

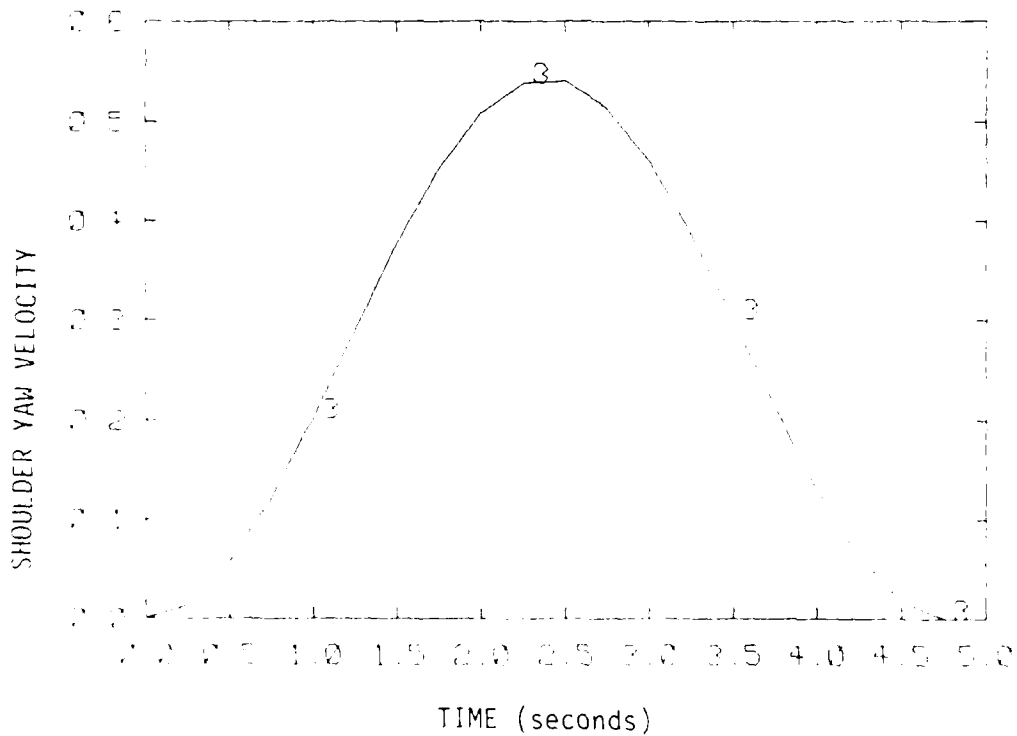


FIGURE 2.22 Shoulder Yaw Velocity - Simulation I

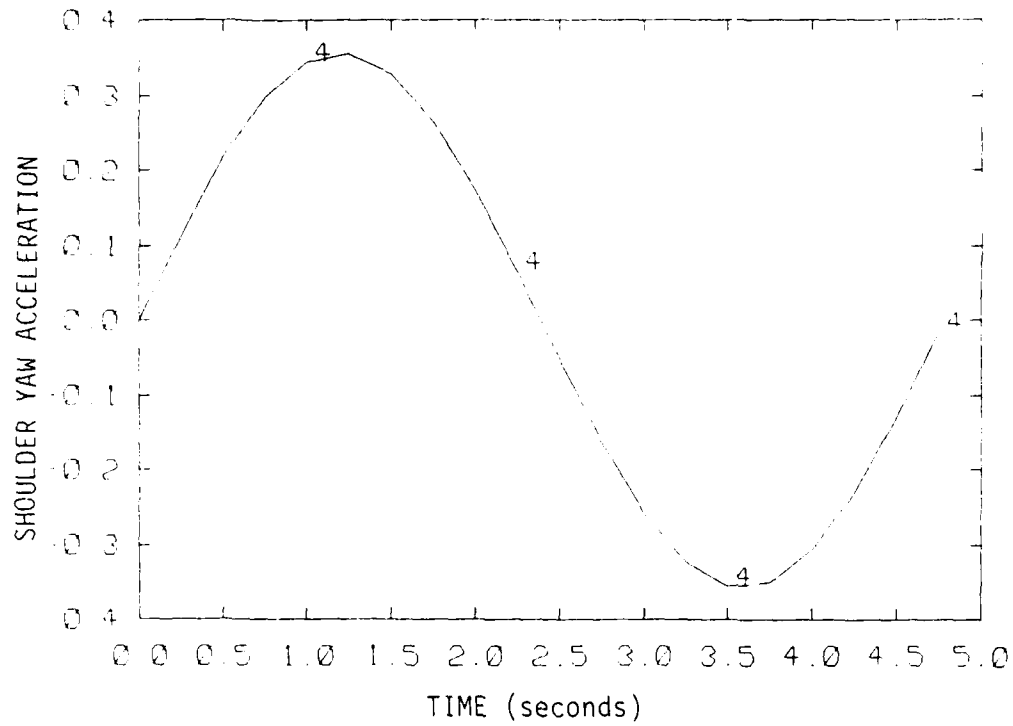


FIGURE 2.23 Shoulder Yaw Acceleration - Simulation I

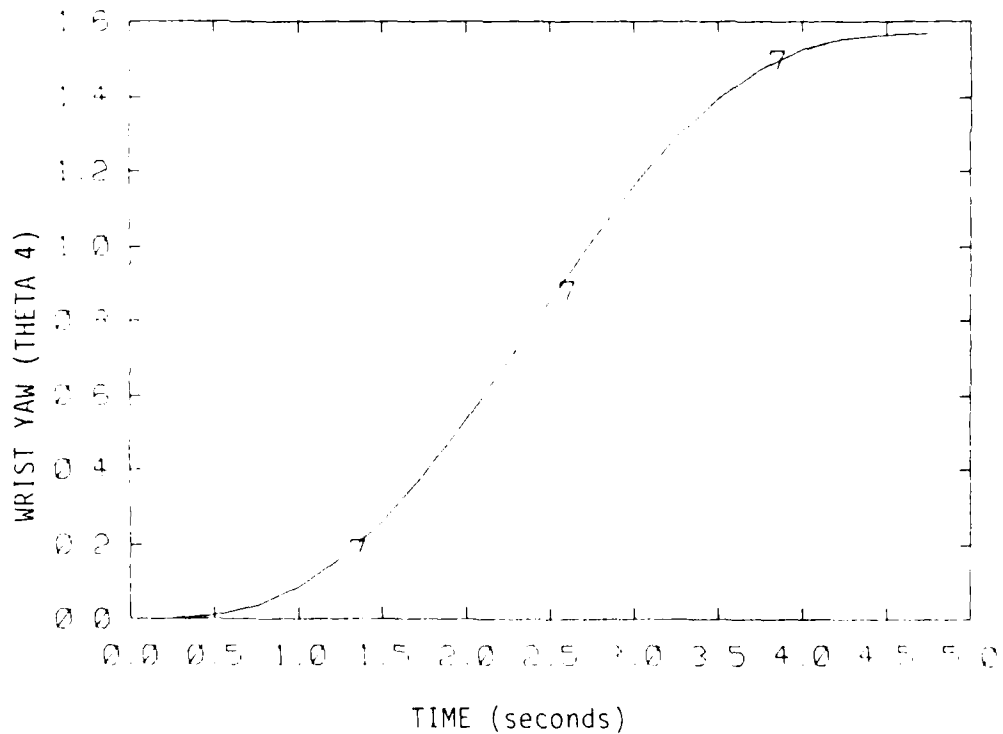


FIGURE 2.24 Wrist Yaw Position - Simulation I

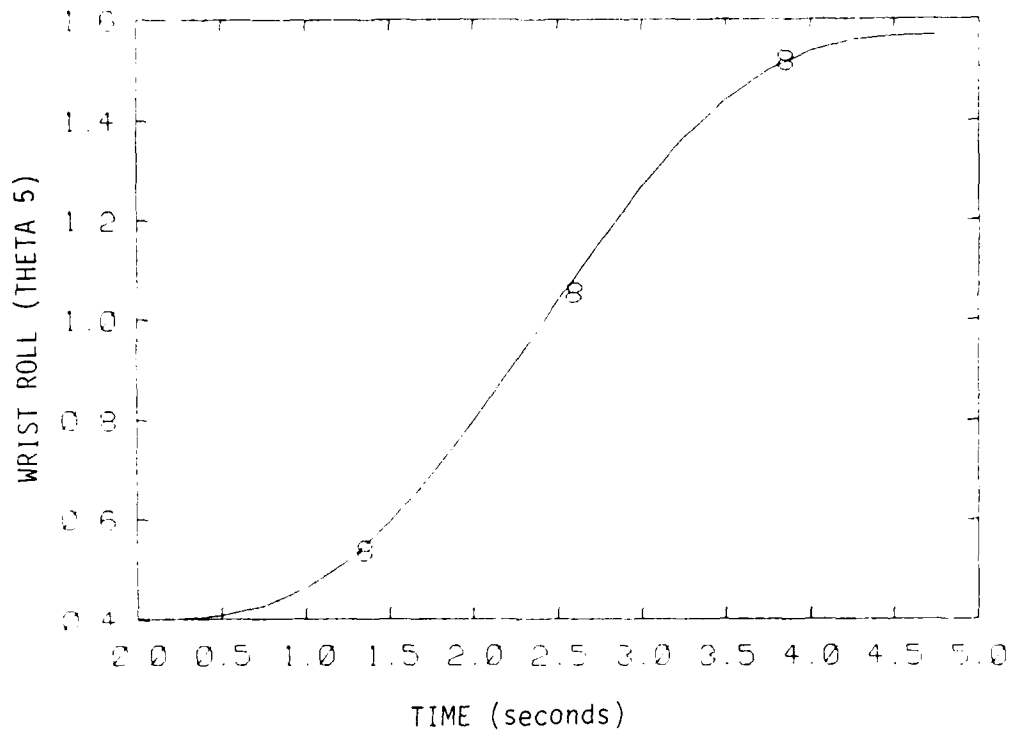


FIGURE 2.25 Wrist Roll Position - Simulation I

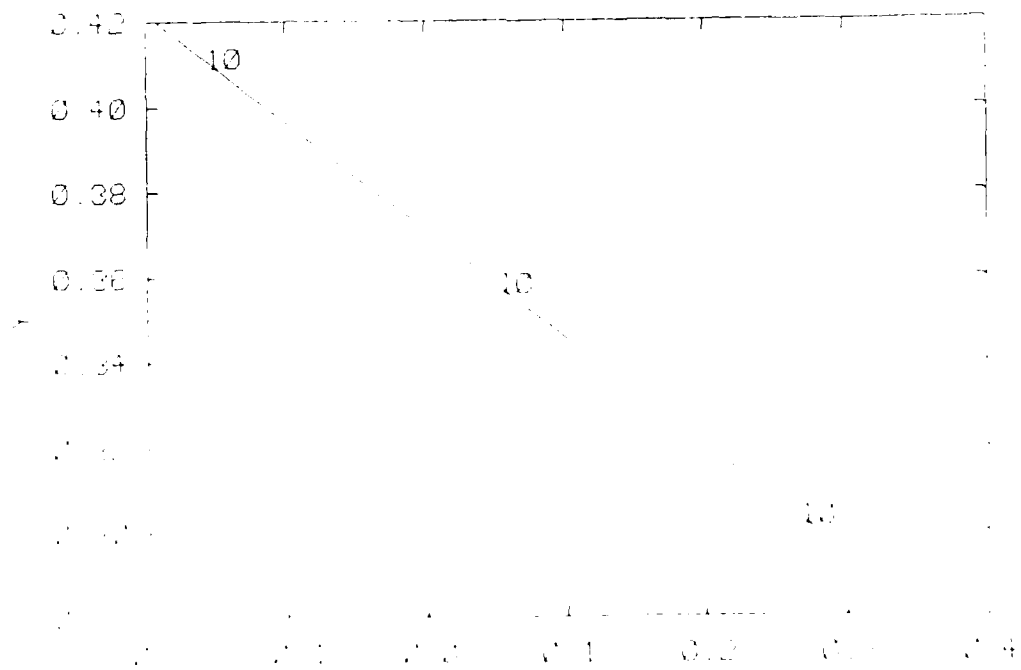


FIGURE 2.26 Y vs. X Position of Pt. A - Simulation I

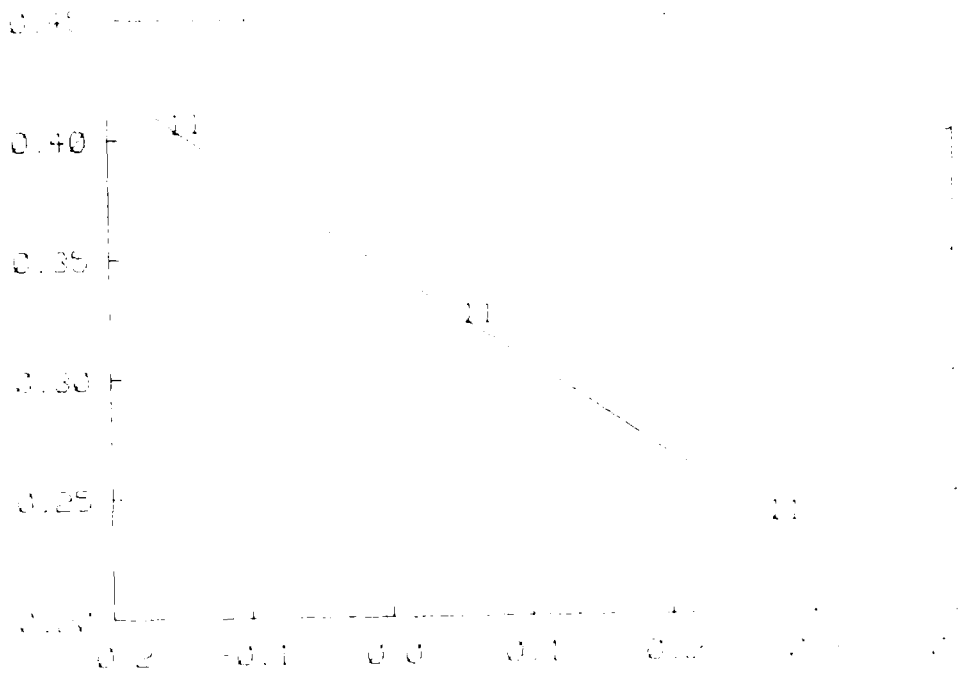


FIGURE 2.27 Z vs. X Position of Point A - Simulation I

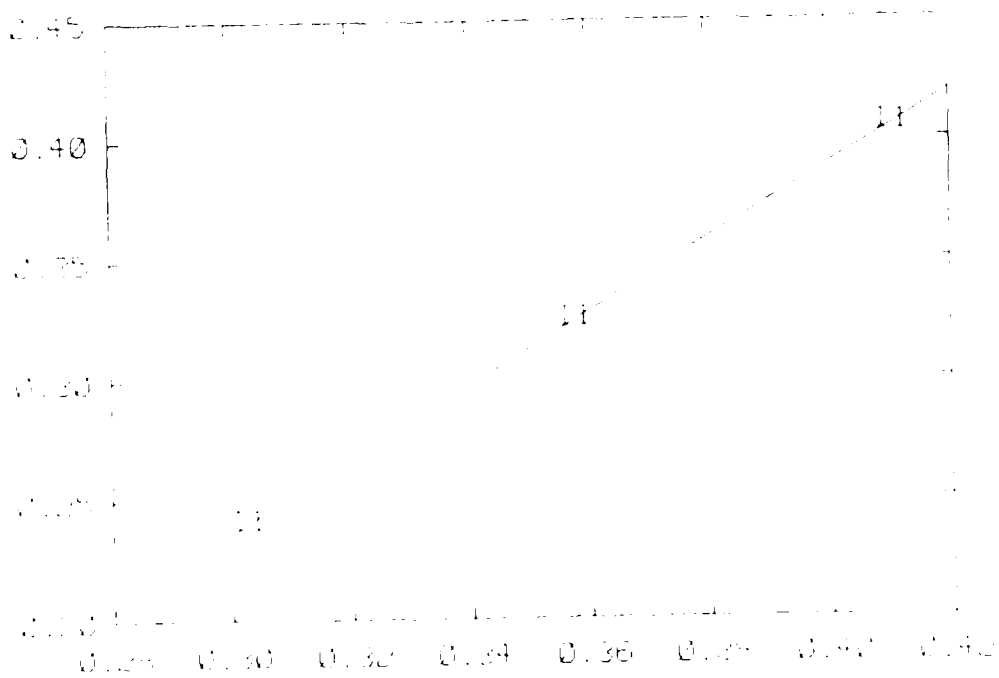


FIGURE 2.28 Z vs. Y Position of Point A - Simulation I

- Simulation 2) - Shoulder yaw initial, θ_{1i} , greater than shoulder yaw final, θ_{1f}
- the elbow is in the elbow up position ($\theta_3 > \pi$).

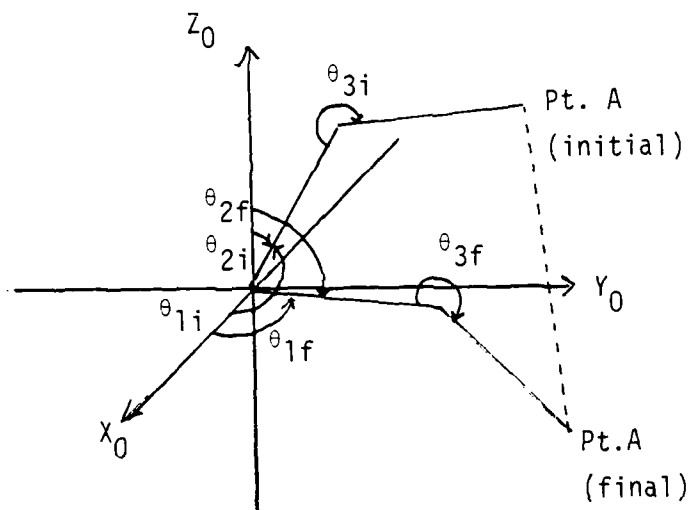


FIGURE 2.29 Arm Position-Simulation II

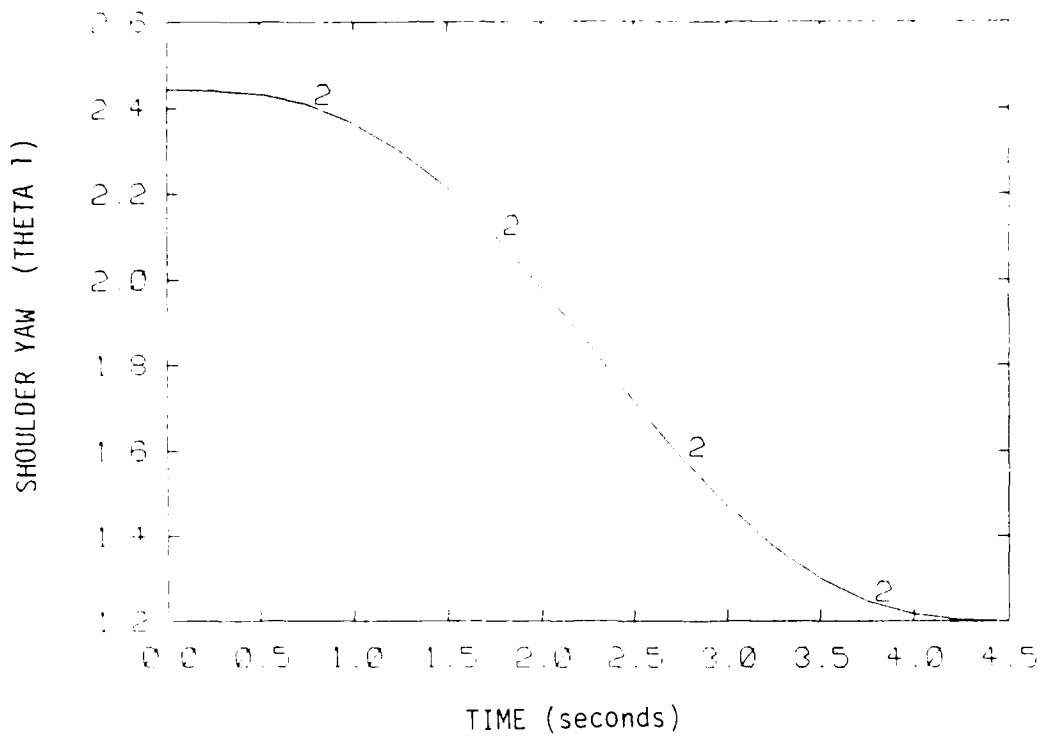


FIGURE 2.30 Shoulder Yaw Position - Simulation II

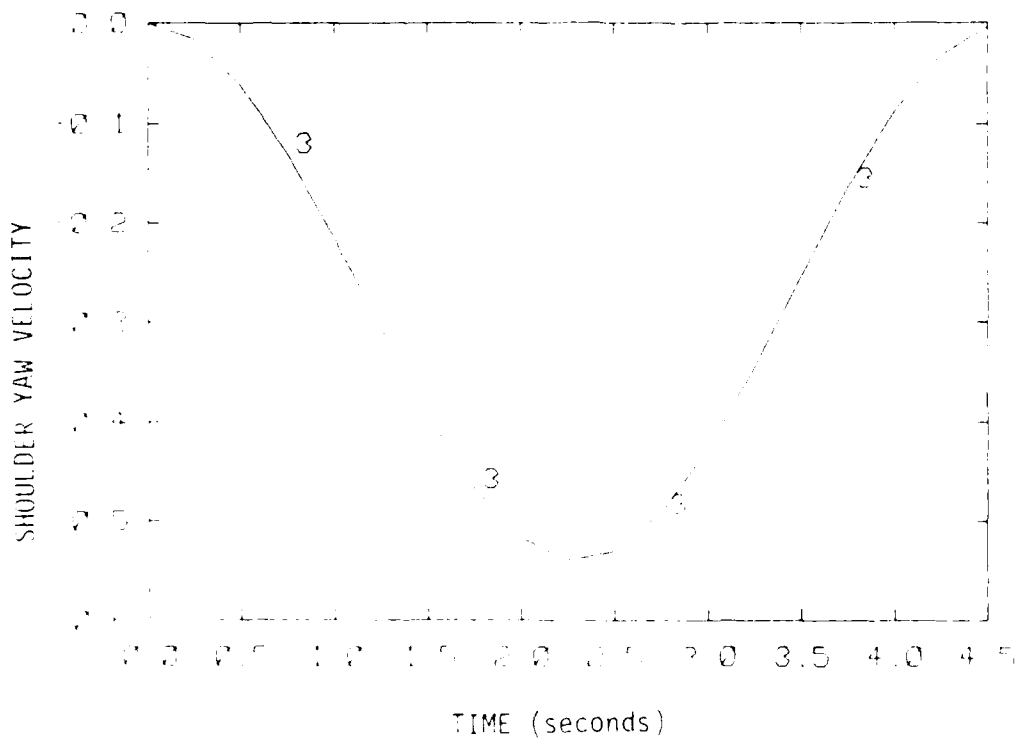


FIGURE 2.31 Shoulder Yaw Velocity - Simulation II

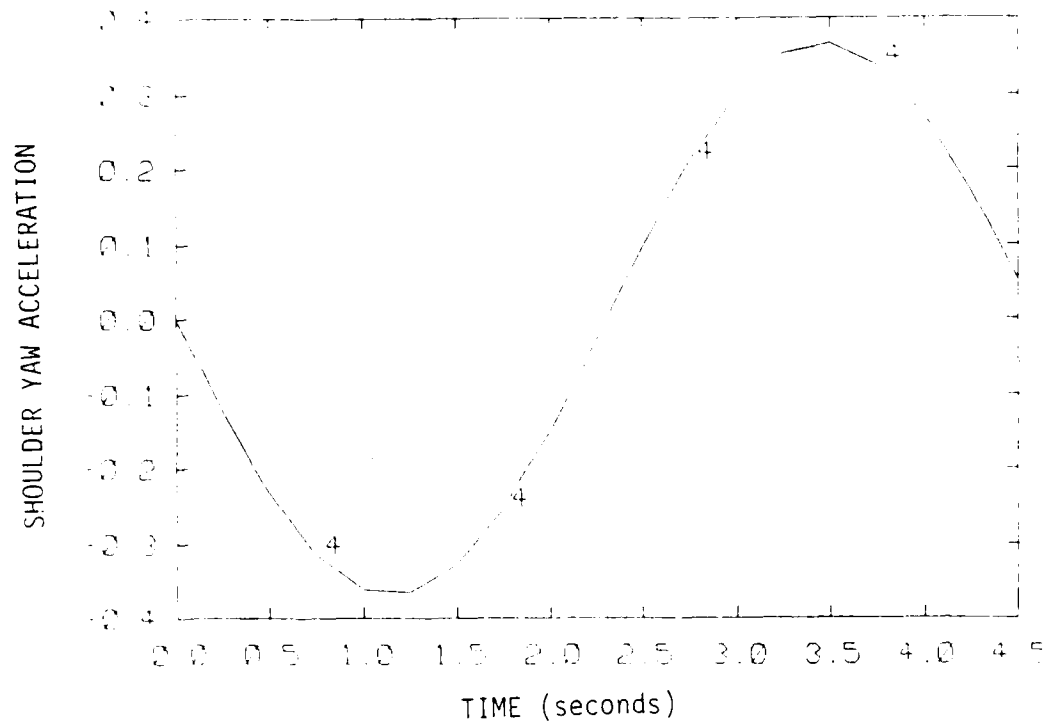


FIGURE 2.32 Shoulder Yaw Acceleration - Simulation II

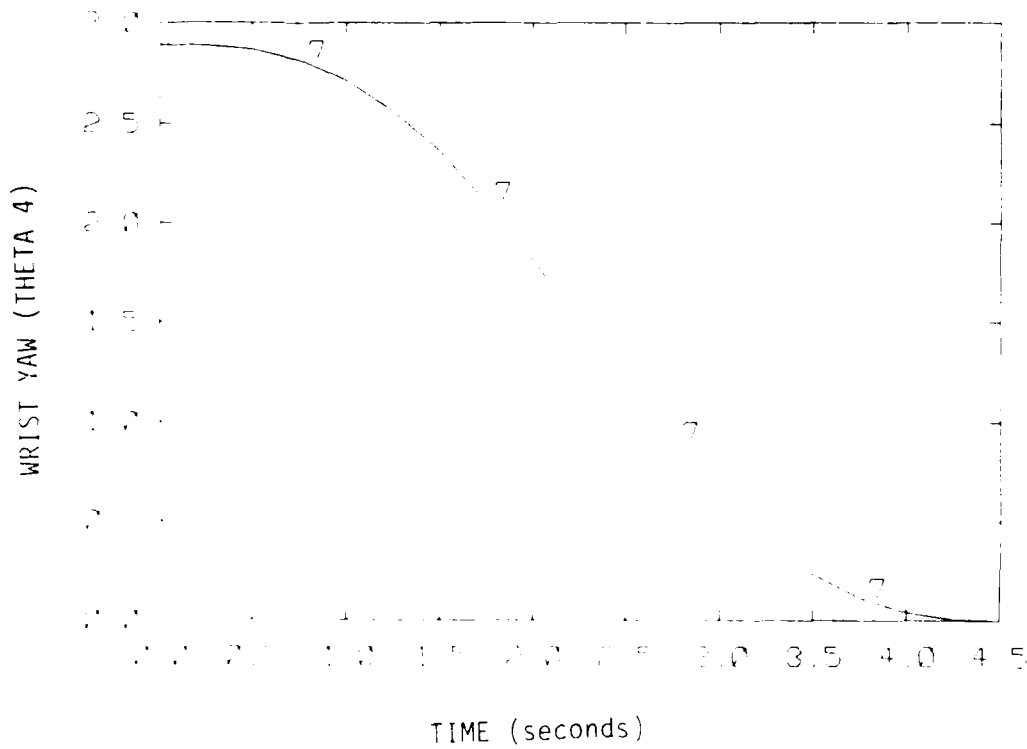


FIGURE 2.33 Wrist Yaw Position - Simulation II

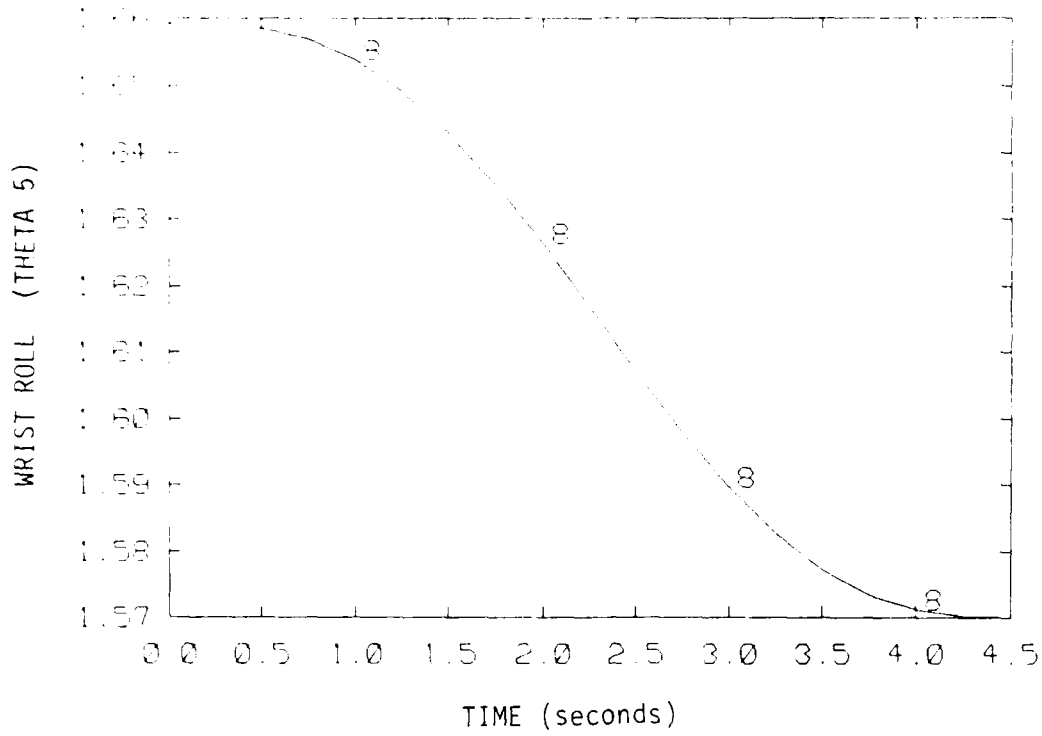


FIGURE 2.34 Wrist Roll Position - Simulation II

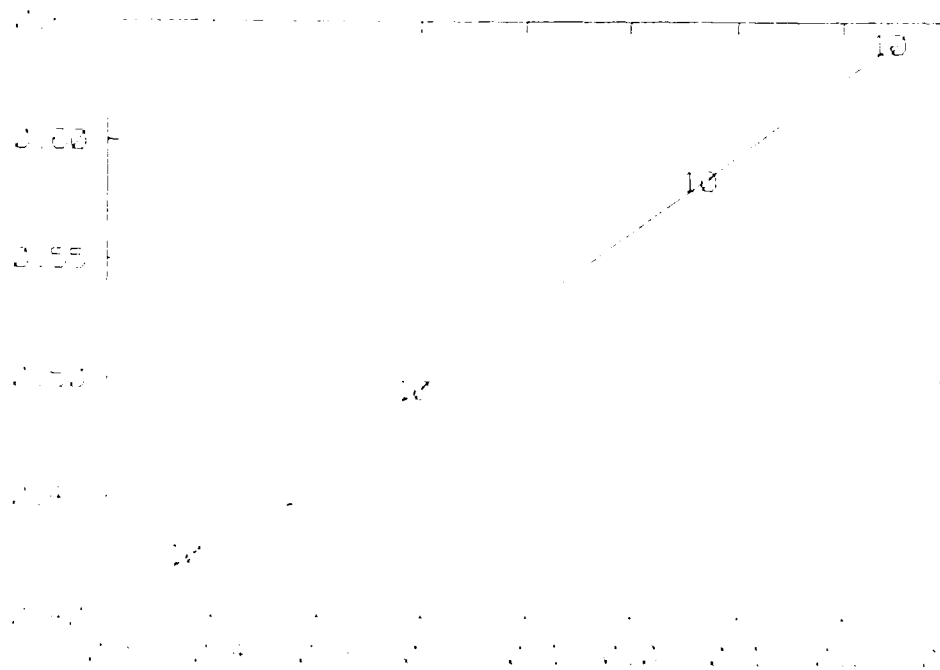


FIGURE 2.35 Y vs. X Position of Pt. A - Simulation II

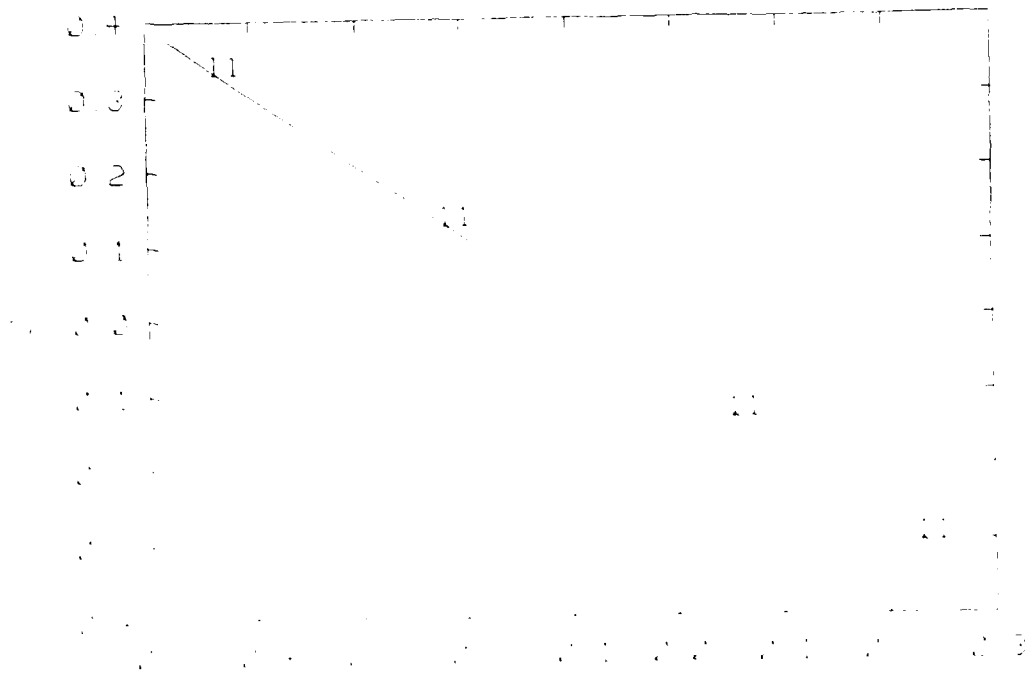


FIGURE 2.36 Z vs. X Position of Pt. A - Simulation II



FIGURE 2.37 Z vs. X Position of Pt. A - Simulation II

- Simulation 3) - Shoulder yaw initial, θ_{1i} , = shoulder yaw final, θ_{1f}
- the elbow is in the elbow down position ($\theta_3 = \pi$)
 - the initial and final orientation angles are equal $\theta_{4i} = \theta_{4f}$, $\theta_{5i} = \theta_{5f}$.

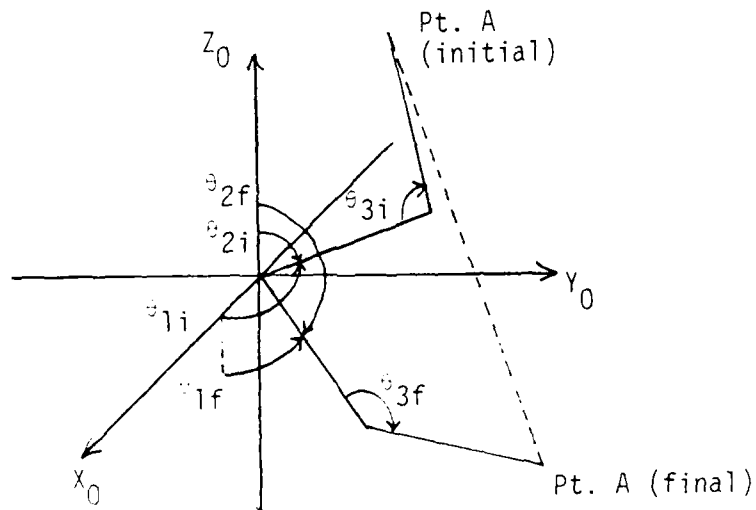


FIGURE 2.38 Arm Position - Simulation III

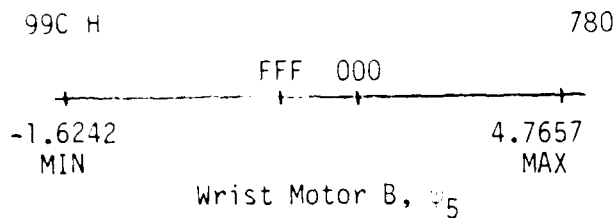
- increment the time step
- synchronize the control system
- start again at the beginning of the loop
until the total time is reached.

When the final point is reached, the function returns control to the calling function. The calling function in this case is the option block. The option used to select this method of control is discussed next.

Using screen functions supplied by the compiler, the operator can select many different options by using a character on the keyboard. These options are outlined in Appendix A (Sect. 3, Part D.5). A new option was added to the option block: if the operator selects 'm', the right arm will move the point A to the specified final point, point B, and the final orientation angles. The new option is outlined below.

- select the supervisory mode
- select option 'm'
- the master arm is read to find the starting configuration of the arm
- the final coordinates of point A and the final orientation angles are recalled from memory
- the move() function is called. The function is passed the initial angular configuration of the arm in integers and the values for the final cylindrical coordinates of point A and the final orientation angles.

- convert the integers of each angle ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$) of the initial position to the appropriate radian value.
- convert the initial position of the arm to cylindrical coordinates (θ, R, Z) of point A and to wrist orientation angle (θ_4, θ_5) [it already has the final cylindrical coordinates of point A and the final orientation angles (θ_4, θ_5).]
- compute the constants necessary for the straight line path constraint
- start the loop
 - using the driving function compute the cylindrical coordinates of point A on the straight line path for that time step
 - using the driving functions for wrist yaw and roll, compute the wrist yaw and roll angles (θ_4, θ_5) for that time step
 - convert the cylindrical coordinates of the point A to joint angles $\theta_1, \theta_2, \theta_3$ (shoulder yaw, shoulder pitch and elbow)
 - check each angle for their limits
 - convert the orientation angles (θ_4, θ_5) to wrist motor angles (ψ_4, ψ_5)
 - convert the angles from radians to integers
 - send the integers for each angle ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$) to the control system



Since the relationship between the angular position and the integer output of the sensor (a linear resistance potentiometer) is a linear one, the conversion from integer to radian is a simple equation. The data from the diagrams shown was used to develop the equations for each angle:

$\theta_1, \theta_2, \theta_3, \psi_4, \psi_5$. Therefore, in this step the central controller uses the existing functions in the BAT software (Appendix A, Sect. 3) to read the master arm. It takes these integer values and converts them to radian values using the equations developed from the data shown in the diagrams above. The central controller now knows the configuration of the master arm in radian measurement of $\theta_1, \theta_2, \theta_3, \psi_4$ and ψ_5 . Here again the assumption is made that the configuration of the master arm is the same as the configuration of the BAT's right arm. These equations are easily inverted, so that they are used to convert from radian measurement to integer value. This is used later in the procedure.

Now, the central controller knows the initial position of the arm and the final point. For this option it calls the straight line path generation function, now called `move()`. The `move()` function is a streamlined version of the simulation program used in the last chapter. The `move()` function is outlined below.

The procedure for obtaining the wrist motor integers was a bit different. Knowing the limits of wrist yaw and wrist roll, the master arm's wrist was moved to these limits and the integers for the wrist motors A and B were recorded.

$$\theta_{4\text{MIN}} = 0.0 \text{ radians}$$

$$\theta_{4\text{MAX}} = 3.14 \text{ radians}$$

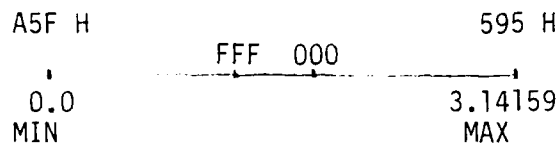
$$\theta_{5\text{MIN}} = -1.74 \text{ radians}$$

$$\theta_{5\text{MAX}} = 1.74 \text{ radians}$$

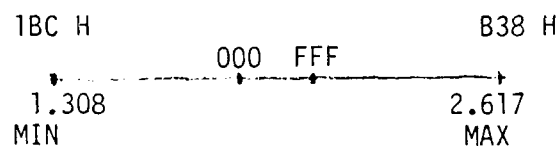
Using the conversion from reference [3] (also in Section 2.3), the corresponding limits for the wrist motors A and B were found. The data for these motors is shown below.



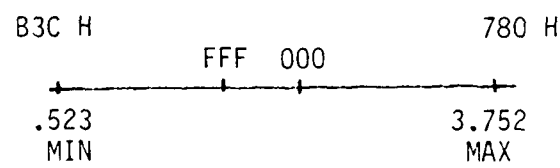
Wrist Motor A, θ_4



Shoulder Yaw, θ_1



Shoulder Pitch, θ_2



Elbow, θ_3

The second step in the procedure is to read the angles of the master arm. Referring to Figure 2.1, the feedback from the right arm on the BAT does not come back to the central controller; the feedback stops at the joint/actuator cards. Therefore, the master arm positions are used to let the central controller know where the right arm of the BAT is. This is also covered in Appendix A. Reading the master arm lets the central controller know the initial angles of shoulder yaw, shoulder pitch, elbow, wrist motor A and wrist motor B. Discussed in Appendix A, Sect. 3, Part D.3, the program uses three functions to read and adjust the analog inputs from the ICS. The five master arm potentiometers are read along with the camera tilt and pan and the left arm roller motor. In this case, the program will read only the five master arm potentiometers. These functions will return integer values for $\theta_1, \theta_2, \theta_3, \theta_4$ and ψ_5 . The integers come from an adjusted analog to digital conversion and represent 12-bit two's complement binary numbers. Using one of the options described in Appendix A, Sect. 3, Part D.5, the software was run with all power to the BAT shut-off. An option was used to show the hexadecimal number read by the central controller when the master arm was moved through each of the joint's limits. This hexadecimal number represented the integer from the adjusted analog to digital conversion. For example, the master arm's shoulder yaw was moved from θ_{1MIN} to θ_{1MAX} and the corresponding numbers were recorded. The three joint angle (shoulder yaw, shoulder pitch and elbow) measurements in radians and hexadecimal numbers are shown below.

In this figure, the operator has grasped two structural components with each arm and would like to make a connection. A supervisory option is selected and the right arm moves autonomously to point B. Point B was chosen to be next to the female end of the left arm's beam. The operator then switches back to master-slave control to make the connection. Point B was measured in cylindrical coordinates, in reference frame X_0, Y_0, Z_0 (Section 2.3), and has the following values:

$\phi = 2.26$ radians (also this is shoulder yaw)

$R = .6$ meters

$Z = -.1$ meters .

The final orientation angles (θ_4, θ_5) were chosen to be:

wrist yaw, $\theta_4 = 1.57$ radians

wrist roll, $\theta_5 = 0.0$ radians .

These values are stored as constants in an array at the beginning of the option. A new position can be measured and the values stored any time prior to the start-up procedure. In the future, an input function could be added to this option, so that the final point can be changed on-line. Presently, to speed up experimentation with this method of control, the final point is stored as a constant in the software.

b) At each time step, the radian measurement for each angle in step 4a) is converted to the appropriate integers for the control system.

c) At each time step, the integers are sent to the control system.

5) Control is returned to the main program.

The final position was chosen to be point B, shown in Figure 3.1.

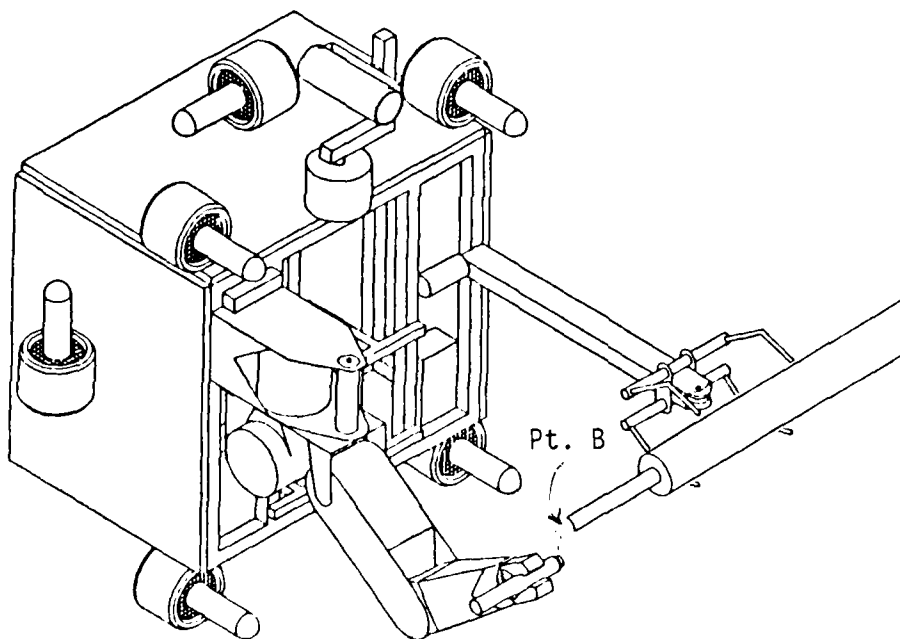


FIGURE 3.1 Finish Point - Implementation I

CHAPTER 3

IMPLEMENTATION I:

UNKNOWN STARTING POSITION, KNOWN FINISH POINT

Using the straight line path generation program developed in the last chapter, supervisory control of the BAT's right arm can be added to the BAT software. With a specified finish point, the operator can select an option using the keyboard and the right arm will move to that point. The procedure is outlined below.

1) The final point A is stored in the program in cylindrical coordinates $(r, R, Z)_f$ along with the final wrist orientation angles $(\theta_4, \theta_5)_f$.

2) The shoulder yaw, shoulder pitch and elbow angles $(\theta_1, \theta_2, \theta_3)_i$ along with the wrist motor A and B angles $(\theta_4, \theta_5)_i$ are read from the master arm potentiometers.

3) The angles read in step 2 are in integers and must be converted to the appropriate angular measurement (radians).

4) The straight line path generation program is called.

a) At each time step, the program generates shoulder yaw, θ_1 , shoulder pitch, θ_2 , elbow, θ_3 , wrist motor A, θ_4 , and wrist motor B, θ_5 angles in radians keeping point A on a straight line path.

This program will be used to implement the supervisory control of the right arm. The first trajectory will be from an unknown starting position to a known finish point and will be discussed in the next chapter.

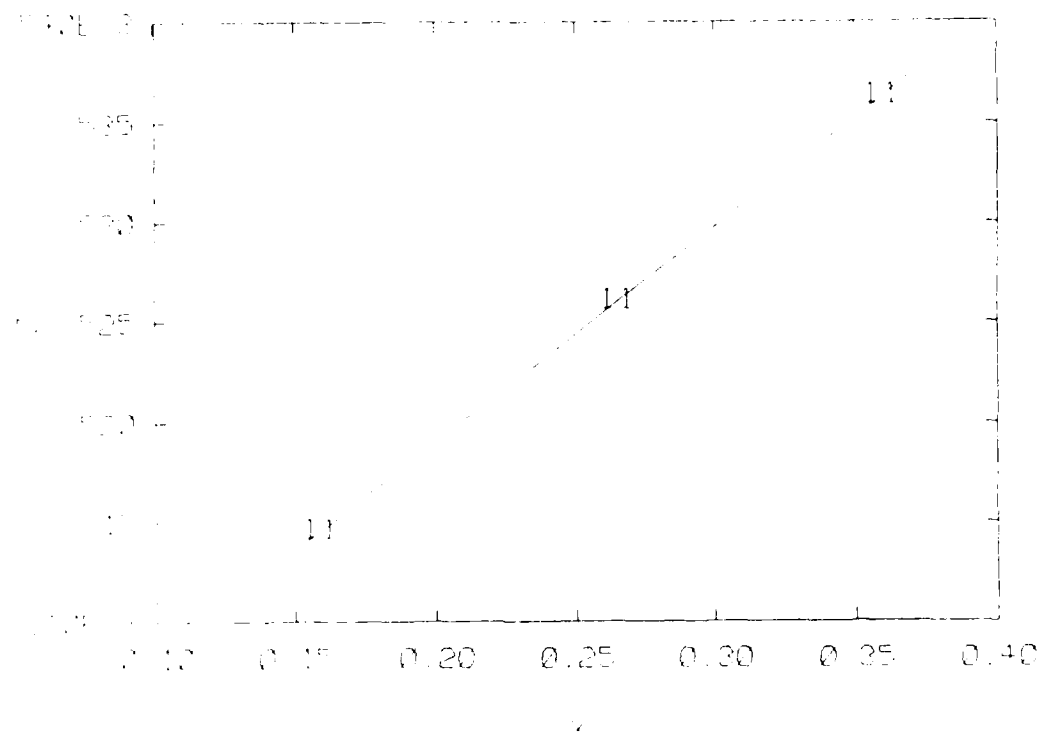


FIGURE 2.45 Z vs. X Position of Point A - Simulation III

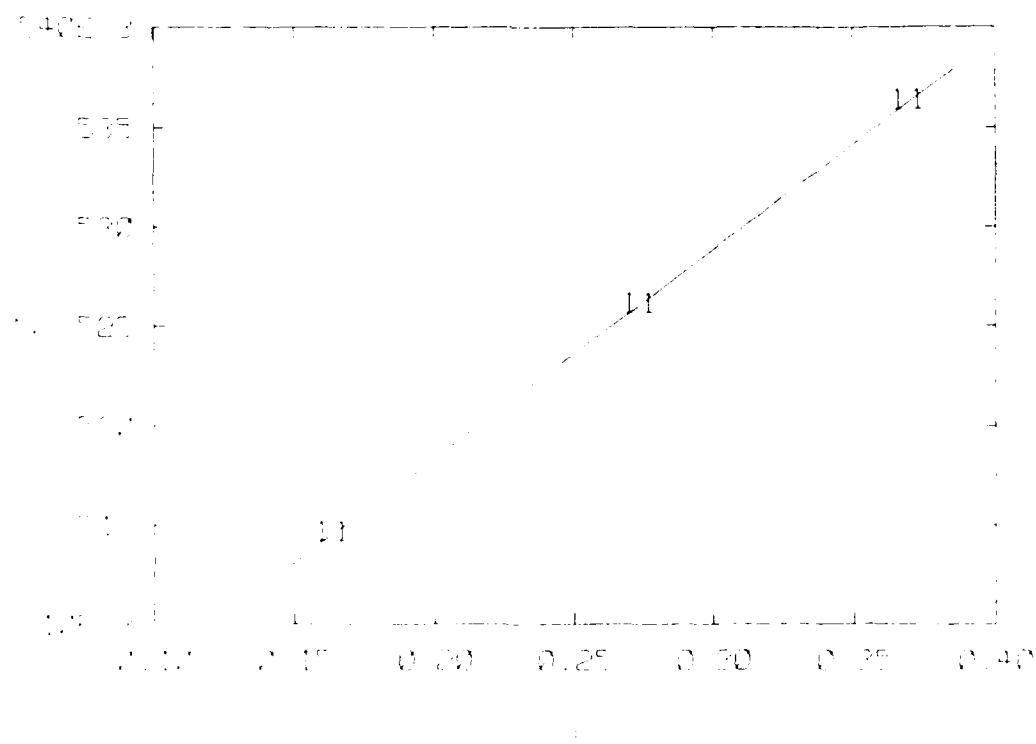


FIGURE 2.46 Z vs. Y Position of Point A - Simulation III

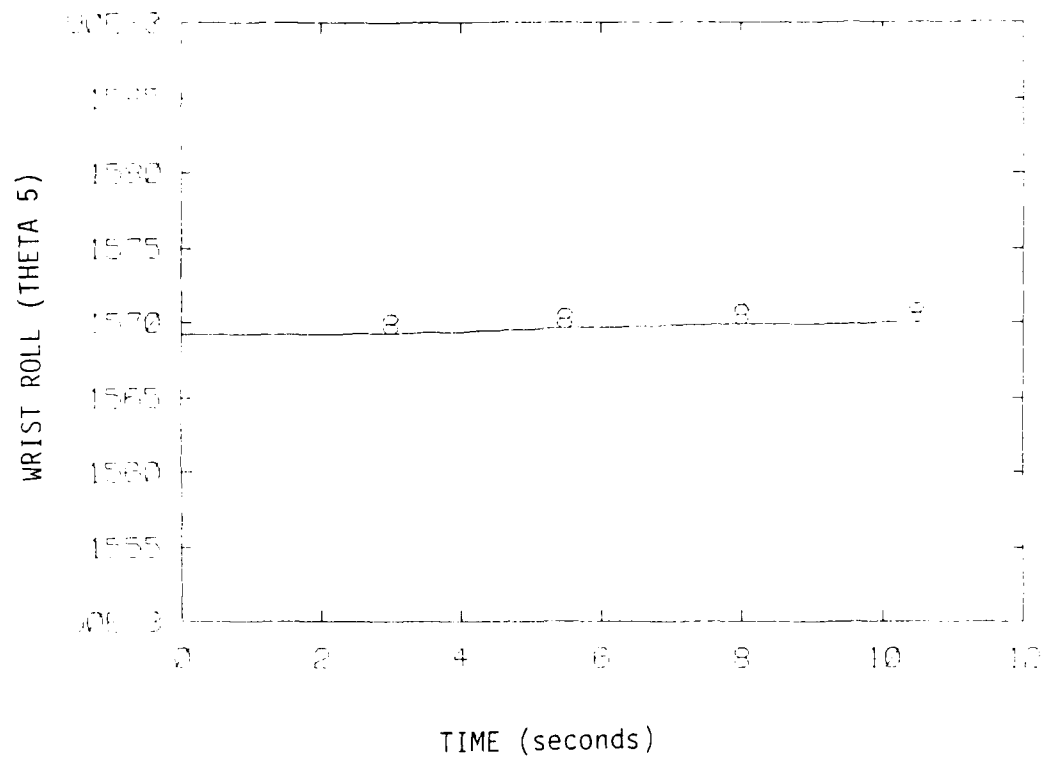


FIGURE 2.43 Wrist Roll Position - Simulation III

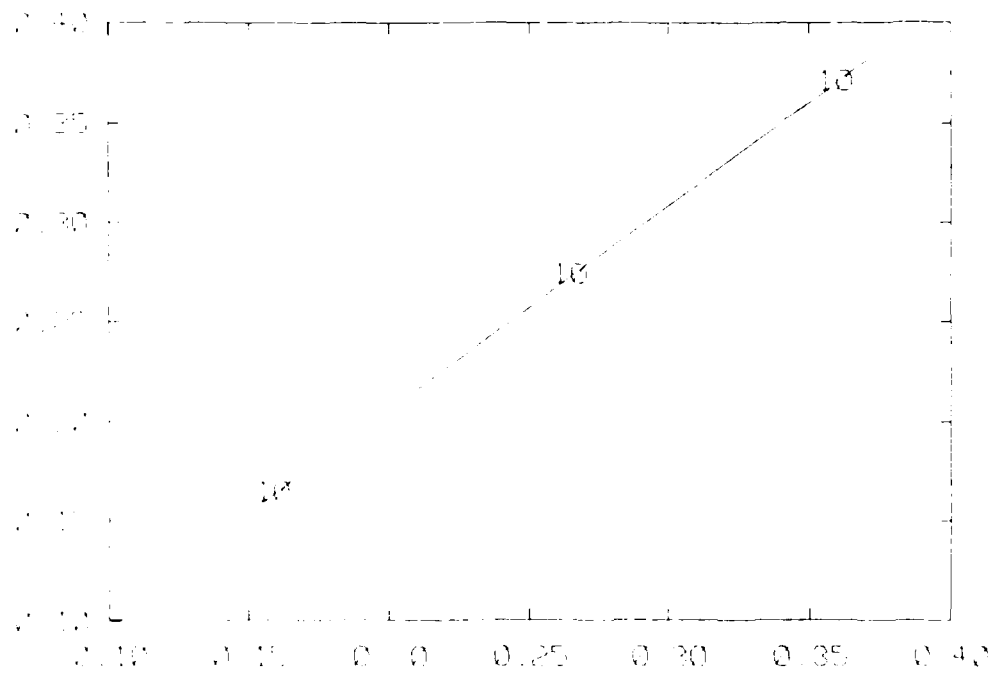


FIGURE 2.44 Y vs. X Position of Point A - Simulation III

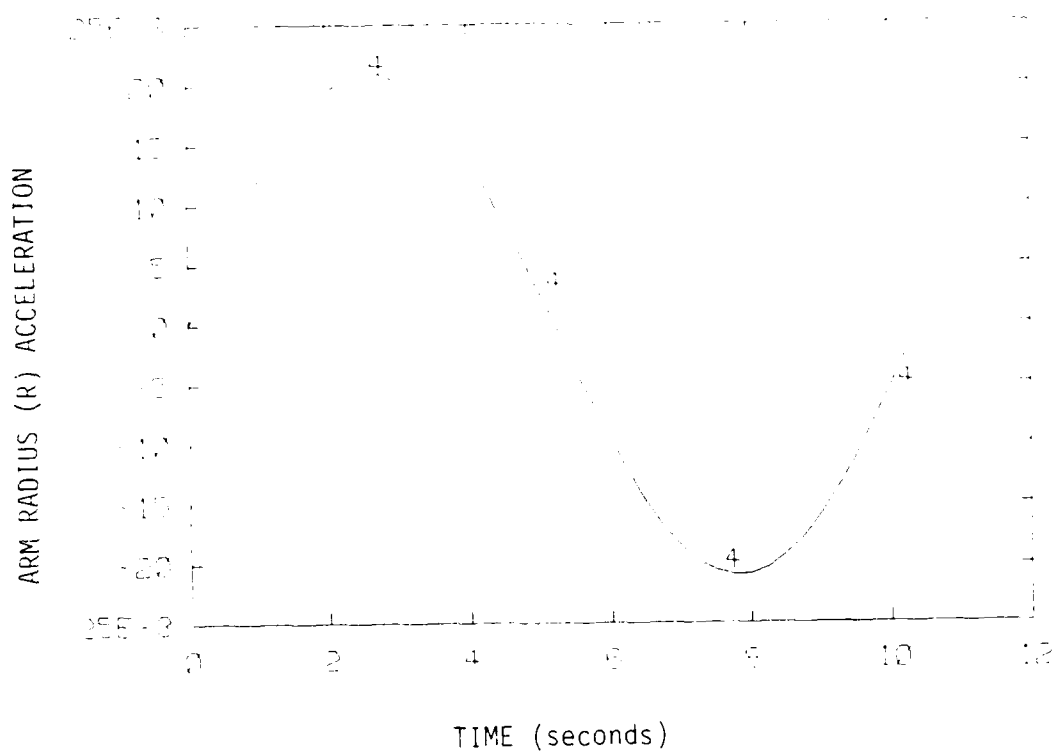


FIGURE 2.41 Arm Radius Acceleration - Simulation III

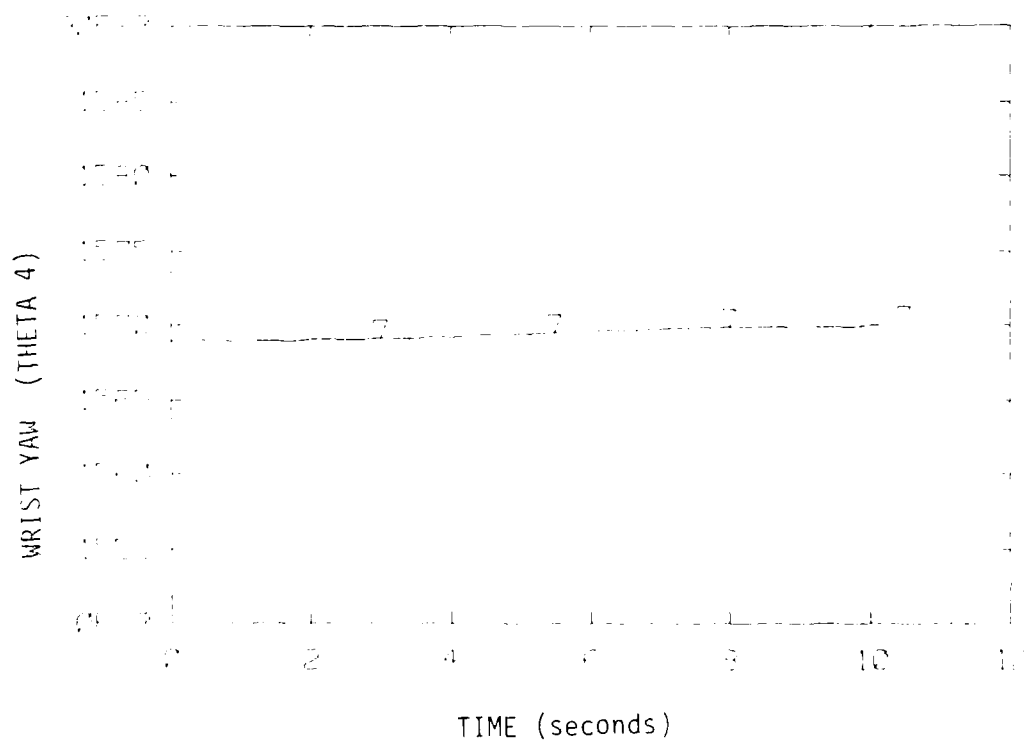


FIGURE 2.42 Wrist Yaw Position - Simulation III

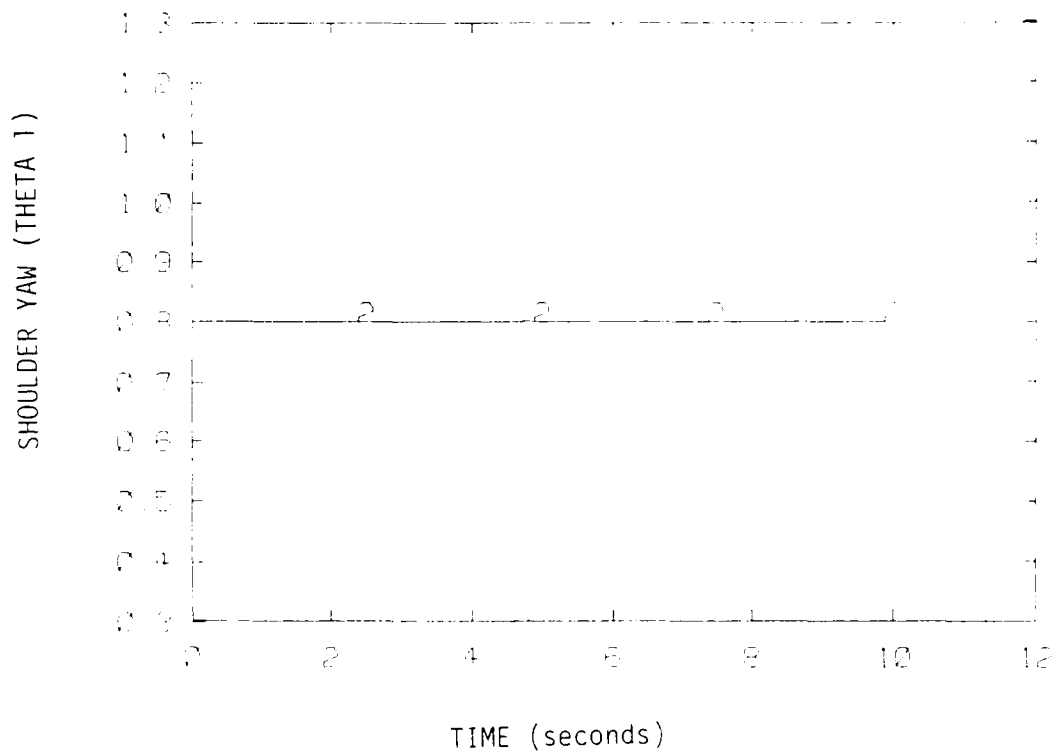


FIGURE 2.39 Shoulder Yaw Position - Simulation III

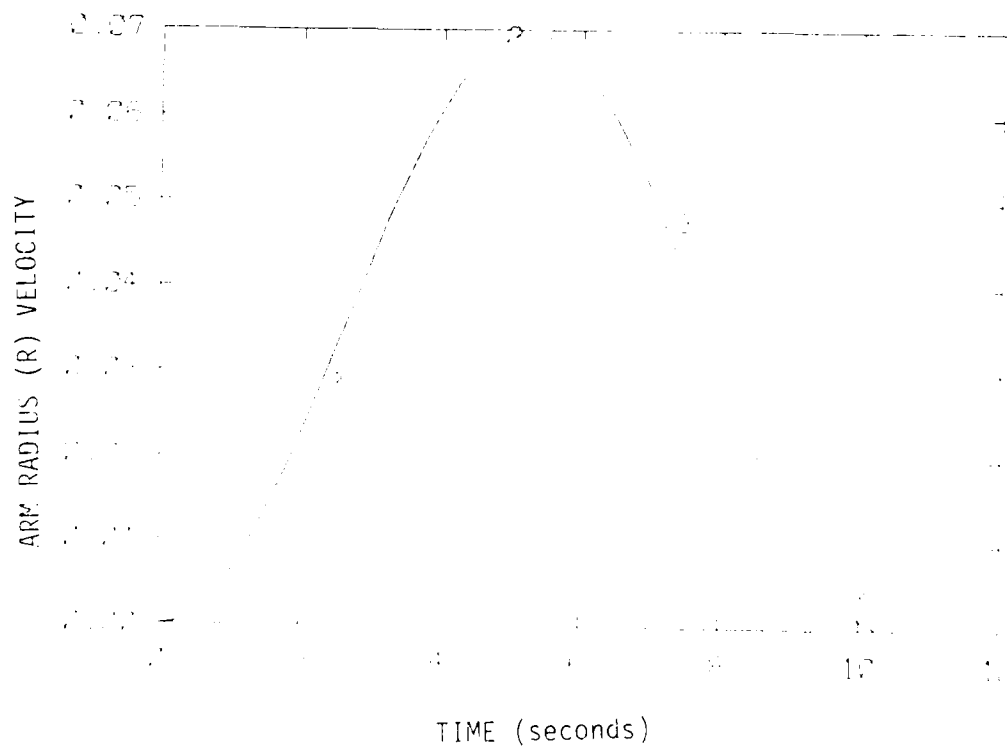


FIGURE 2.40 Arm Radius Velocity - Simulation III

- after the arm has completed the move, exit the supervisory mode.

The space bar is used as an emergency exit from both this option and the supervisory mode.

The program was tested in two steps:

- step 1) select the option, enter the initial arm configuration in integers instead of using the master arm, write the joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$) to a file instead of sending them to the control system. This step checks the program flow and the integer/radian conversions.

In step one, the following integers (shown with their corresponding radian value) were entered:

Shoulder yaw, θ_1 = .785 corresponds to 3373.

Shoulder pitch, θ_2 = 1.57 radians corresponds to 110

Elbow, θ_3 = .785 radians corresponds to 3130

Wrist yaw, θ_4 = 0.0 radians

Wrist roll, θ_5 = 1.57 radians

these correspond to:

Wrist motor A, θ_4 = -1.46 radians corresponds to 4064

Wrist motor B, θ_5 = -1.46 radians corresponds to 2550.

The arm was initially configured as shown in Figure 3.2. The resulting plots follow.

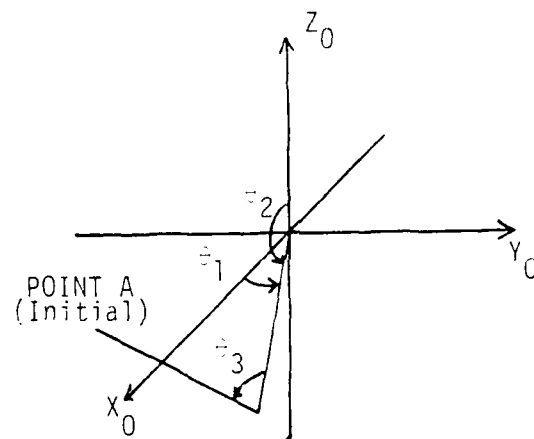


FIGURE 3.2 Arm Position - Step 1, Implementation 1

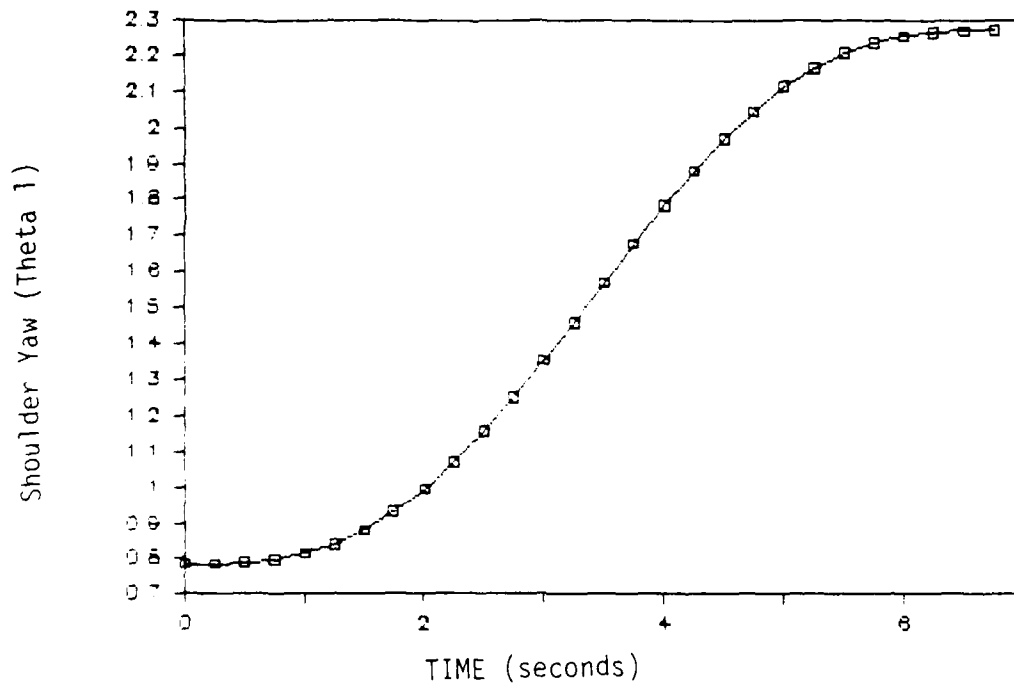


FIGURE 3.3 Shoulder Yaw Position, Step 1, Implementation I

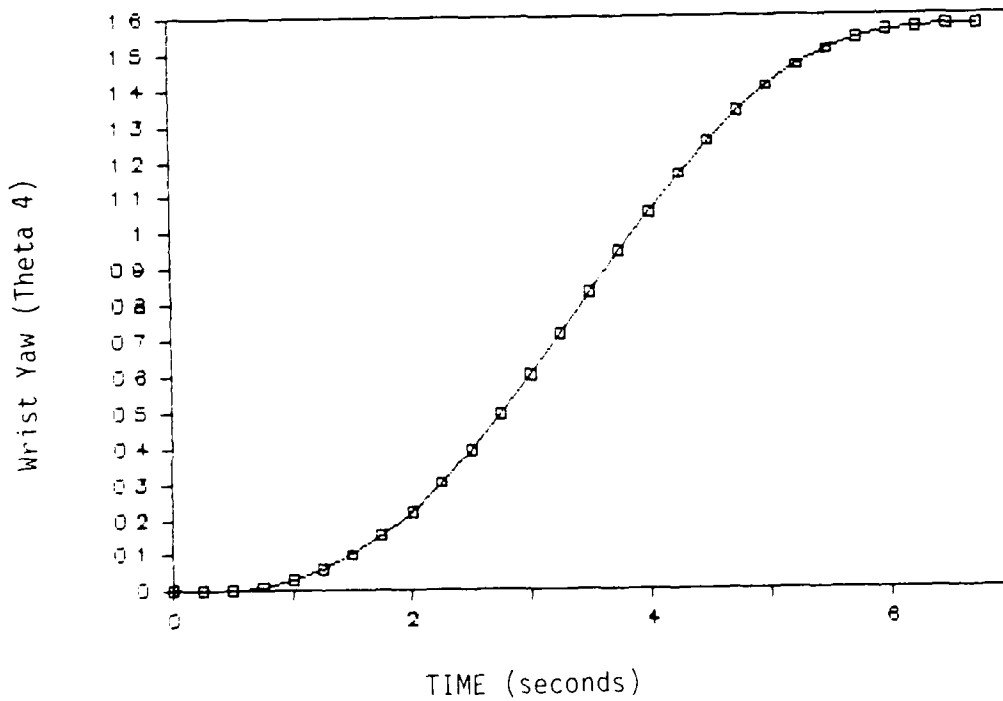


FIGURE 3.4 Wrist Yaw Position, Step 1, Implementation I

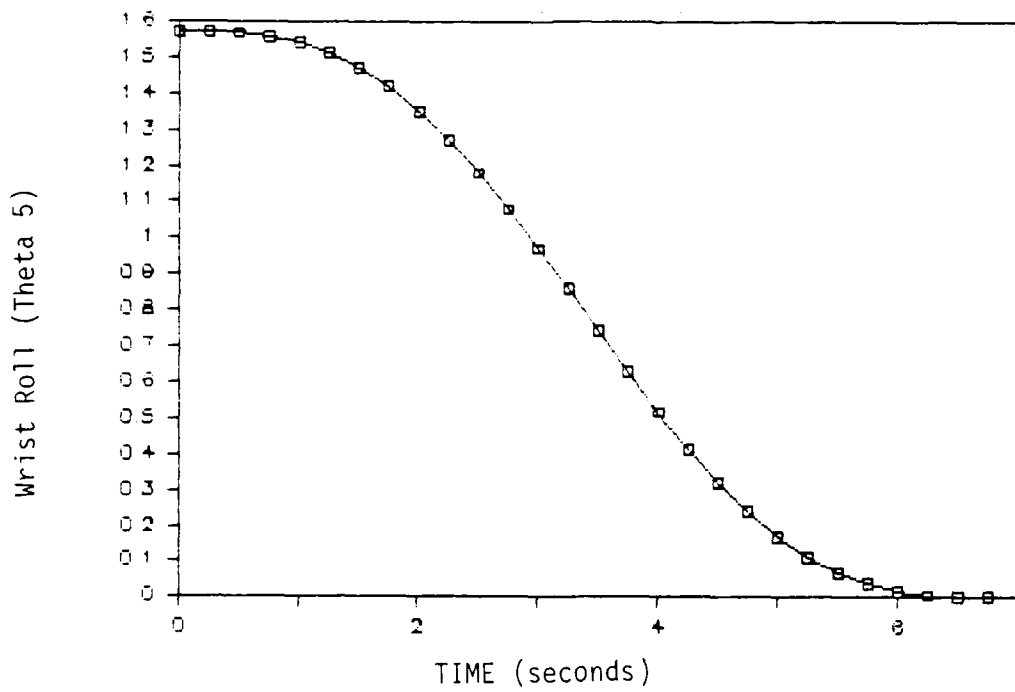


FIGURE 3.5 Wrist Roll Position, Step 1, Implementation I

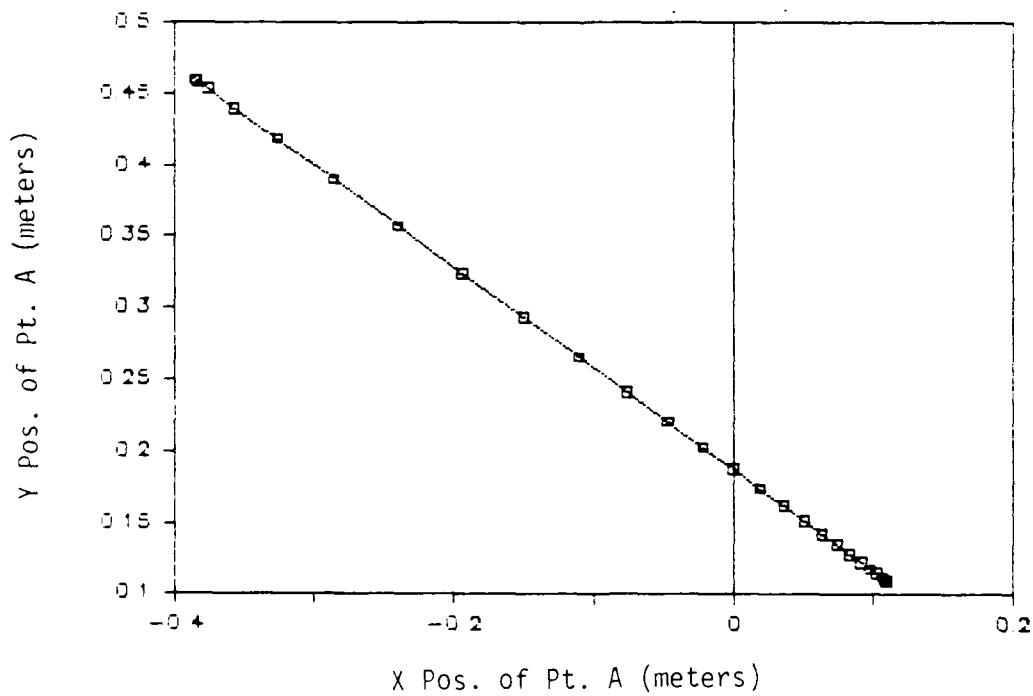


FIGURE 3.6 Y vs. X Position, Step 1, Implementation II

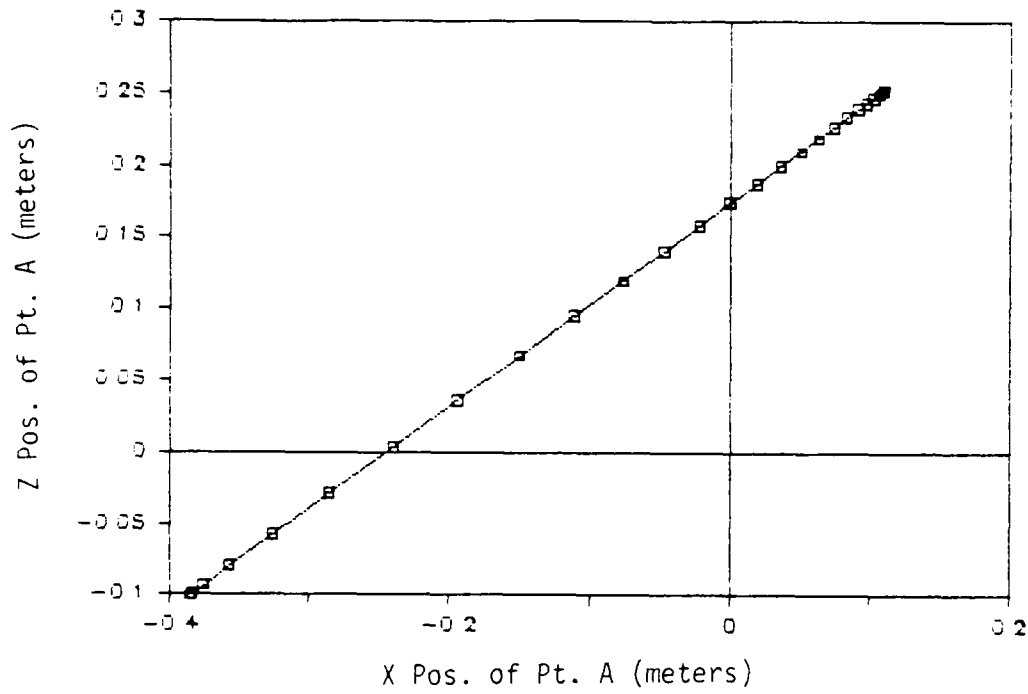


FIGURE 3.7 Z vs. X Position, Step 1, Implementation I

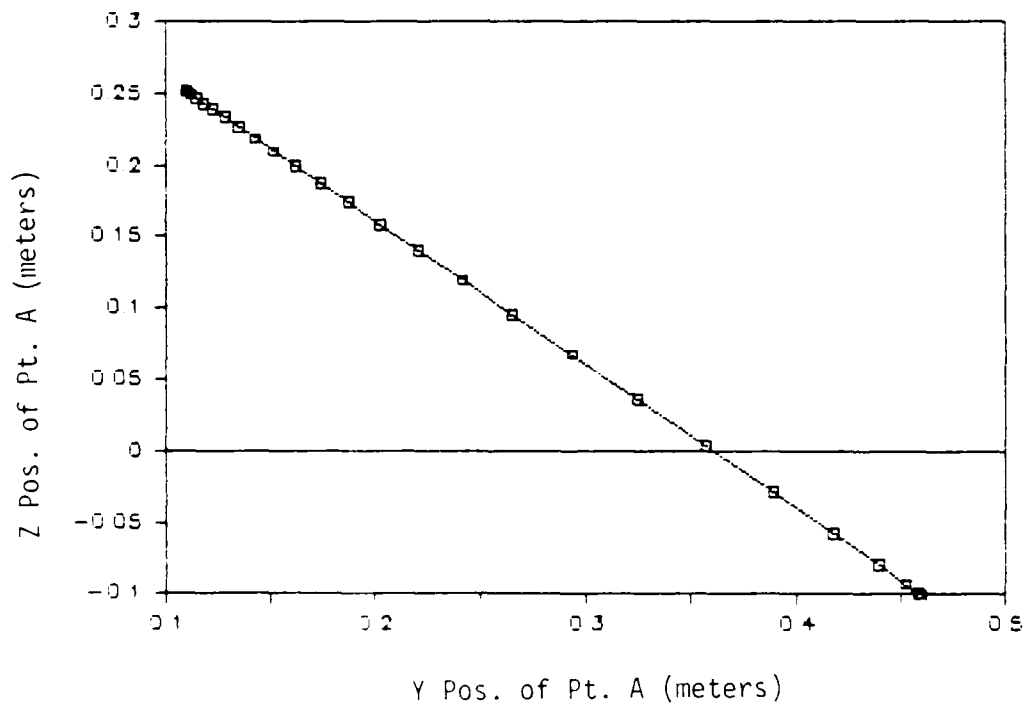


FIGURE 3.8 Z vs. Y Position, Step 1, Implementation I

As shown by the plots, there is a gradual acceleration and gradual braking of the shoulder yaw, wrist yaw and wrist roll angles. The last three plots show how the program does generate a straight line path for point A and it does finish at point B.

Step 2) Select the option, send the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ to a file instead of to the control system. This step checks everything except the actual movement of the BAT's right arm.

This step was run at the ICS and the values obtained are shown in the plots which follow.

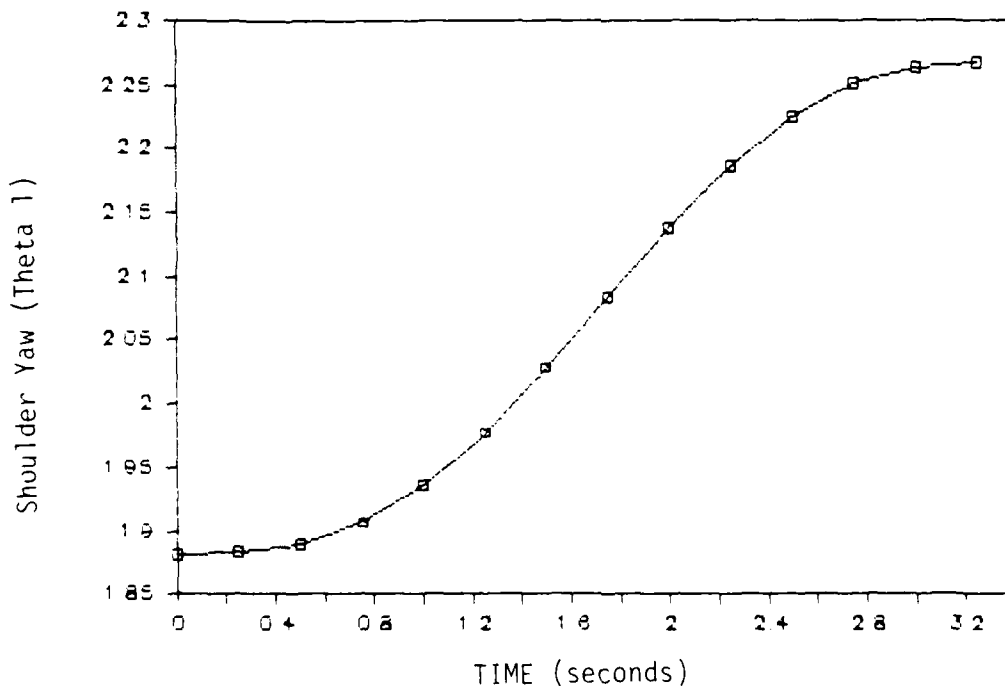


FIGURE 3.9 Shoulder Yaw Position, Step 2, Implementation I

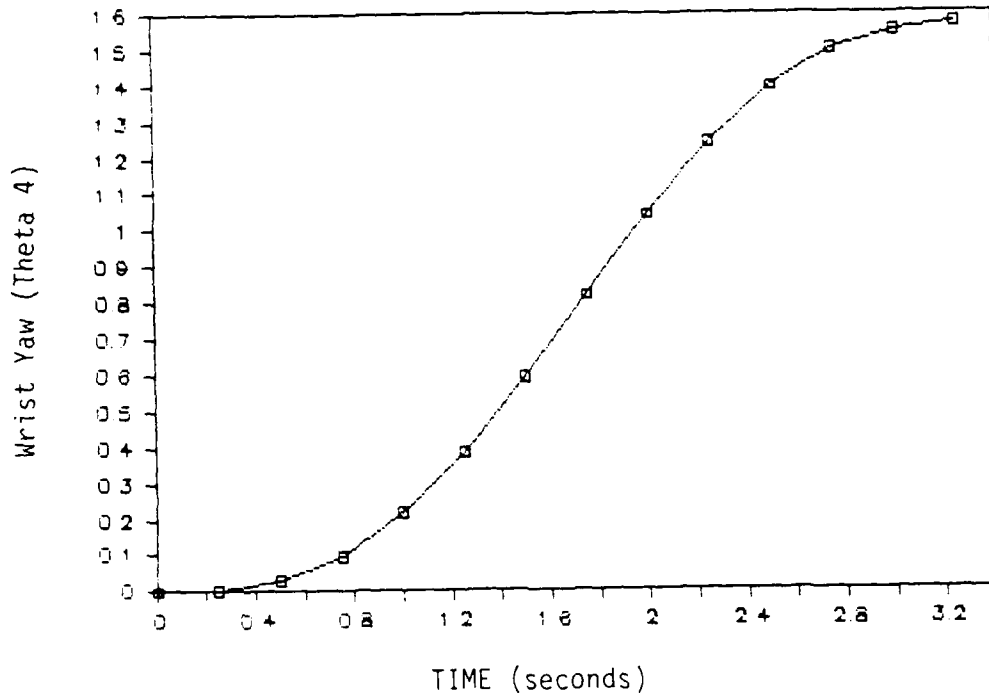


FIGURE 3.10 Wrist Yaw Position, Step 2, Implementation I

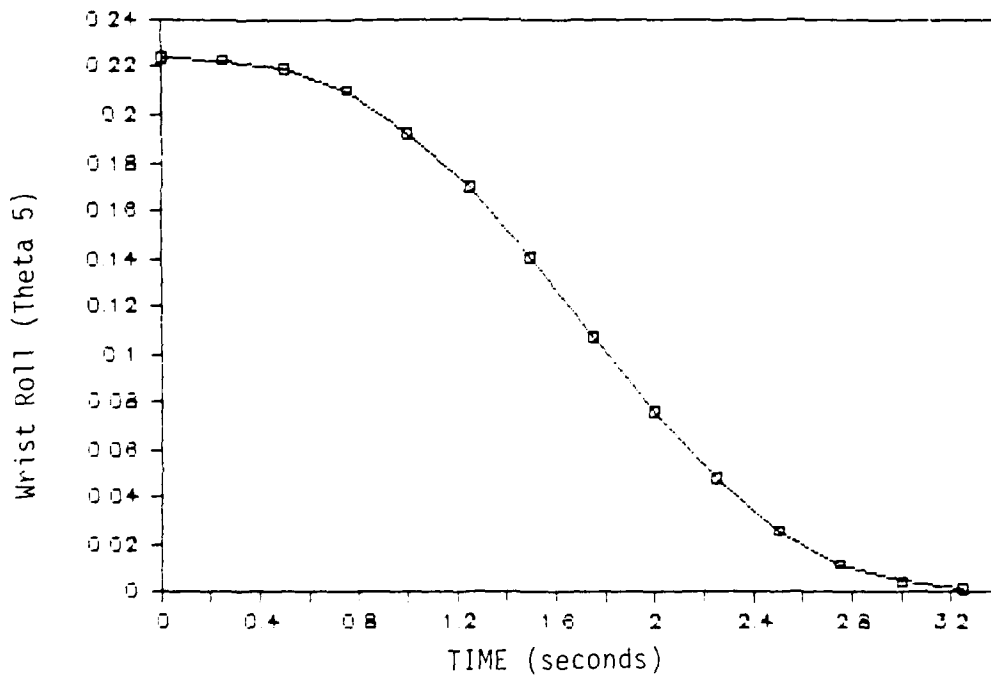


FIGURE 3.11 Wrist Roll Position, Step 2, Implementation I

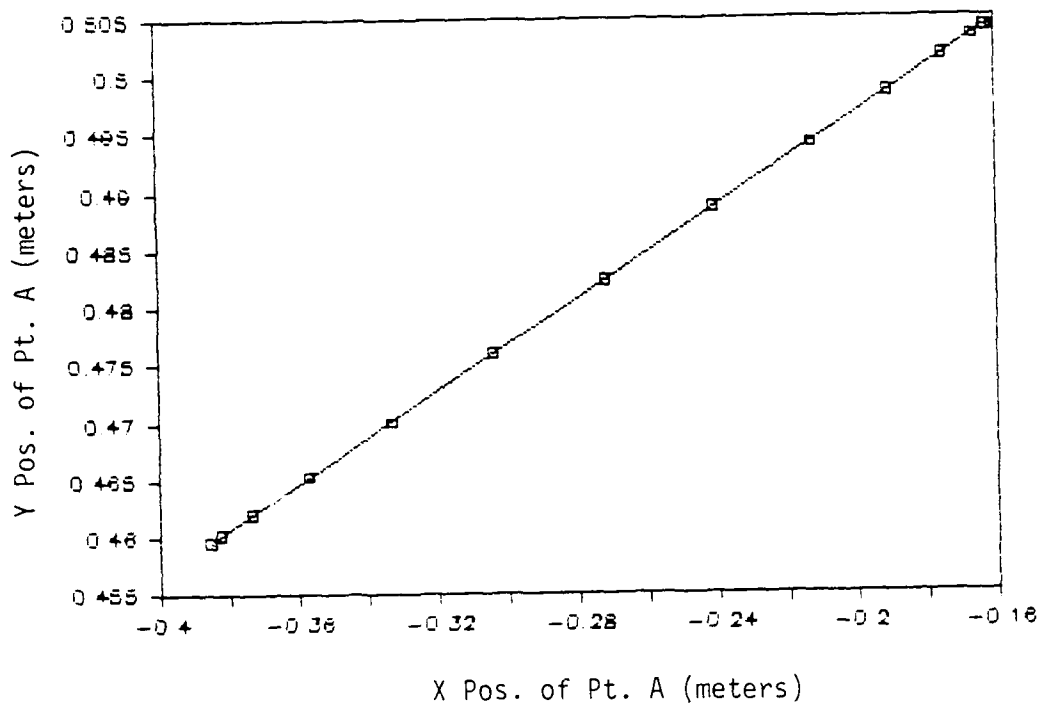


FIGURE 3.12 Y vs. X Position, Step 2, Implementation I

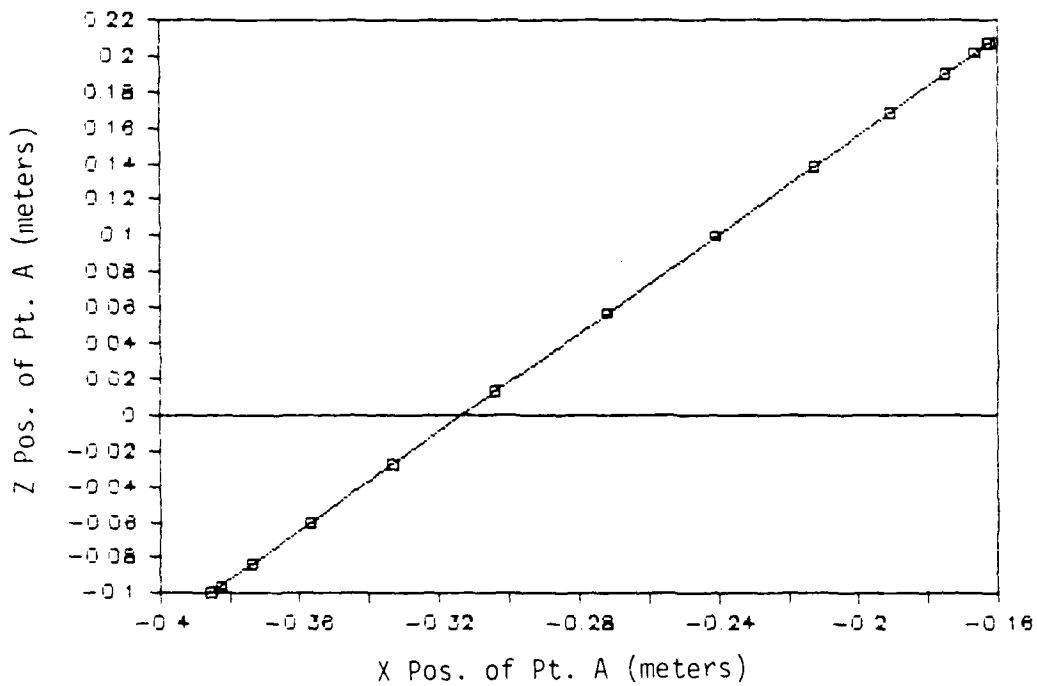


FIGURE 3.13 Z vs. X Position, Step 2, Implementation I

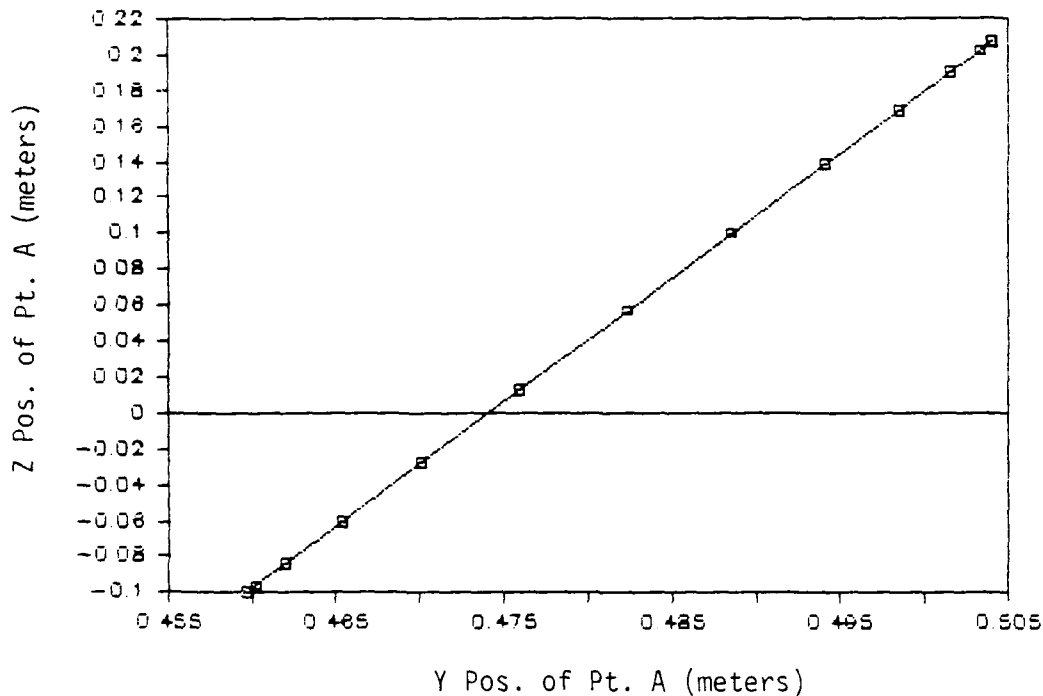


FIGURE 3.14 Z vs. Y Position, Step 2, Implementation II

The plots show how the angles (shoulder yaw, wrist yaw, and wrist roll) are gradually accelerated and gradually braked. The last three plots demonstrate the straight line generation of point A.

Actual tests done on the BAT with this method of control will be discussed in the last chapter. This chapter shows that the right arm control system can be upgraded to supervisory control. This now gives the operator the capability of moving the right arm to a specified position without using the master arm.

CHAPTER 4
IMPLEMENTATION II:
LEARNING TRAJECTORIES BY CONCATENATION

The last chapter demonstrated that the central controller could control the right arm of the BAT, by moving the right arm's end effector in a straight line path from any point to any other point, within the arm's range and flexibility limits. An improvement on this capability would be to enable the central controller to remember a sequence of points, concatenate their paths and move the right arm along that specified trajectory. This can be done easily with the `move()` function developed in the last chapter and the addition of three new options in the software.

The first new option enables the central controller to remember a sequence of points in space. These are the points that the operator would like the path of the right arm's end effector to move through. It can be done by reading the position of the master arm and storing that position in an array. This operation consists of four steps.

- 1) enter the supervisory mode
- 2) choose this option by hitting the 'x' on the keyboard
- 3) the central controller reads the five potentiometers of the master arm corresponding to the shoulder yaw, θ_1 , shoulder pitch, θ_2 , elbow, θ_2 , wrist motor A, ψ_4 , and

wrist motor B, ψ_5 . It stores these integers in a two-dimensional array (7 rows by 5 columns) with each column corresponding to $\theta_1, \theta_2, \theta_3, \psi_4, \psi_5$ (the integer value) in that order. The entire first row is skipped. This row is saved for the starting point. An operator can store as many as five points. The last row is saved for the finish point which is the same point as the starting point. Obviously, each row is a right arm position.

4) return to the calling function.

The maximum number of points stored can be increased easily by making the array have more rows. For this thesis it will remain at five.

The second new option allows the operator to clear the array of all points and start over. When the 'y' key is selected, the option simply re-initializes the array's elements to all zeroes. It then returns to the calling function.

Now the operator has the capability of storing any points in the right arm's range of motion. The operator simply moves the right arm, using the master arm, to a point and selects 'x'. The point is stored in a row of integers corresponding to the master arm's potentiometer readings of the joint angles. The operator then goes to the next point and selects 'x' again. This point is then stored in the row below the first stored point. The operator does this until all the points desired have been stored. The software joins these points in sequence to

make a desired trajectory. (See Figure 4.1.) The operator has the capability of clearing the trajectory and starting over. This is used when entering another trajectory or when a mistake has been made in entering one of the points.

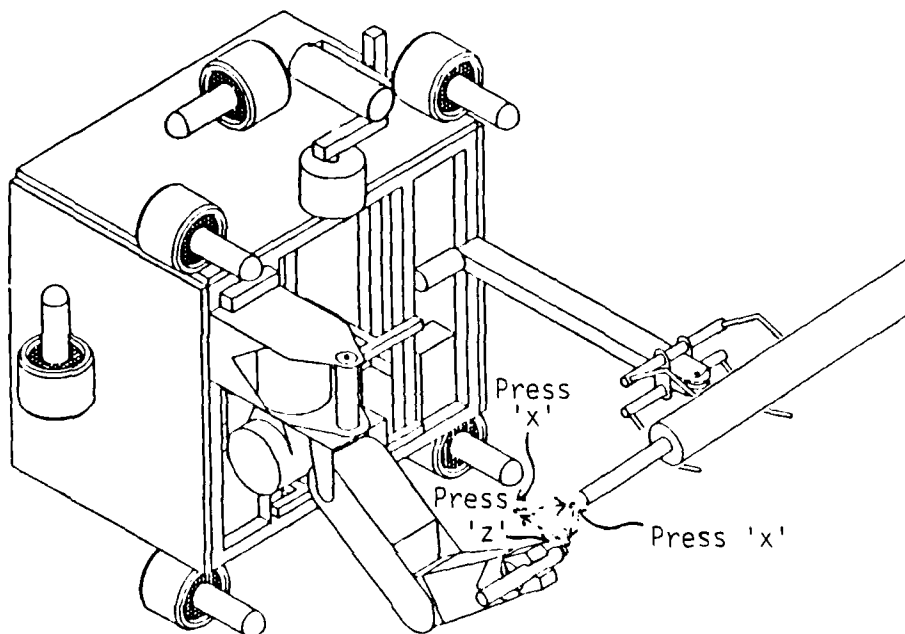


FIGURE 4.1 Implementation II

The final option is the one which will move the right arm from wherever it is, through the sequence of points which were stored and then move back to where it originally started. The paths through all these points will be a straight line. The outlines of this new option is listed below.

- 1) enter the supervisory mode
- 2) choose to move through all the stored points by selecting 'z' on the keyboard
- 3) read the current position of the master arm and store this position in the array as the first row and directly after the last row stored (for example, if only three points have been stored when 'z' is selected the current position is in row number 1 and row number 5, row numbers 2,3,4 are the three points already stored by selecting 'x')
- 4) start a loop in which the program takes two rows of integers at a time, starting with the first and second row, and set these points (a row of integers) equal to the initial and final positions for the straight line path
- 5) convert the integers of the final position to cylindrical coordinates of point A and final wrist orientation angles ($\theta, R, Z, \theta_4, \theta_5$). Of course, to do this it must first convert the five integers to shoulder yaw and pitch, elbow, and wrist motor angles to radians. It then uses the radian values of θ_1, θ_2 and θ_3 to convert to θ, R, Z of point A and θ_4, θ_5 to convert to θ_4, θ_5 . This was covered in Chapter 2.
- 6) call the `move()` function passing it to the initial position, which is the first row of angles in integers, and

the final point A and orientation angles, found in step 5. the move() function moves the end effector of the right arm (point A) from the initial position to the final point in a straight line path.

7) when the move() function is completed, control goes back to step 4 and takes the next set of two points (in this case rows two and three, the next time it will take rows three and four and so on) and repeats steps 5, 6 and 7

8) when the last pair is used the right arm will be back at the starting point (this is because the current position was also stored as the last point in the array)

9) control is returned to the calling function.

In testing this method of concatenating points the same two step procedure used in testing the method of Chapter 3 was used here. First, all entries were made as integers by keyboard input and all angles were sent to a file. This tested program flow and integer/radian inversion. Second, the master arm was used to enter the points and the values were sent to a file. This tested all aspects of the program except the actual movement of the BAT's right arm. Actual tests done on the BAT will be discussed in Chapter 5.

AD-A156 856

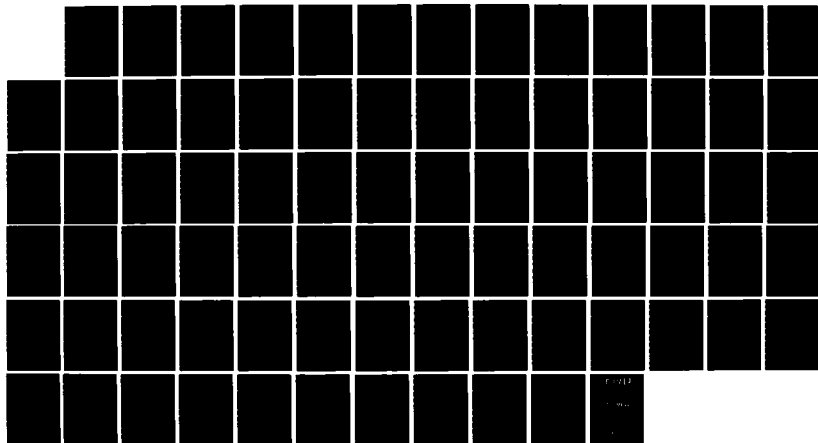
SUPERVISORY CONTROL OF THE RIGHT ARM OF THE BEAM
ASSEMBLY TELEOPERATOR(U) ARMY MILITARY PERSONNEL CENTER
ALEXANDRIA VA A J HANGANIELLO 10 MAY 85

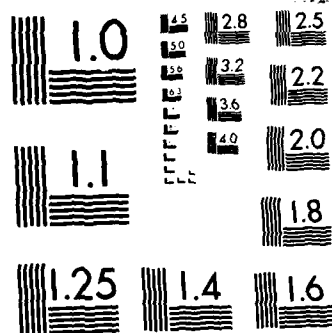
2/2

UNCLASSIFIED

F/G 13/9

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Step 1) Keyboard entry, values sent to a file (see Figure 4.2).

	START/FINISH	POINT 1	POINT 2	POINT 3
Shoulder yaw	.785 rad 3373	1.5707 rad 4090	1.570 rad 4090	1.22 rad 3771
Shoulder pitch	1.570 rad 110	1.91986 rad 3662	1.919 rad 3662	2.18 rad 3428
Elbow	.785 rad 3130	.78539 rad 3130	1.57 rad 3894	1.047 rad 3385
Wrist yaw	1.570	0.0 rad	.785 rad	2.35 rad
Wrist roll	0.0 rad	1.570 rad	.785 rad	0.0 rad
Wrist motor A	-1.57 rad 4002	-1.462 rad 4064	-1.516 rad 4033	-2.35 rad 3553
Wrist motor B	1.570 rad 141	-1.462 rad 2550	.054 rad 3394	2.35 rad 579

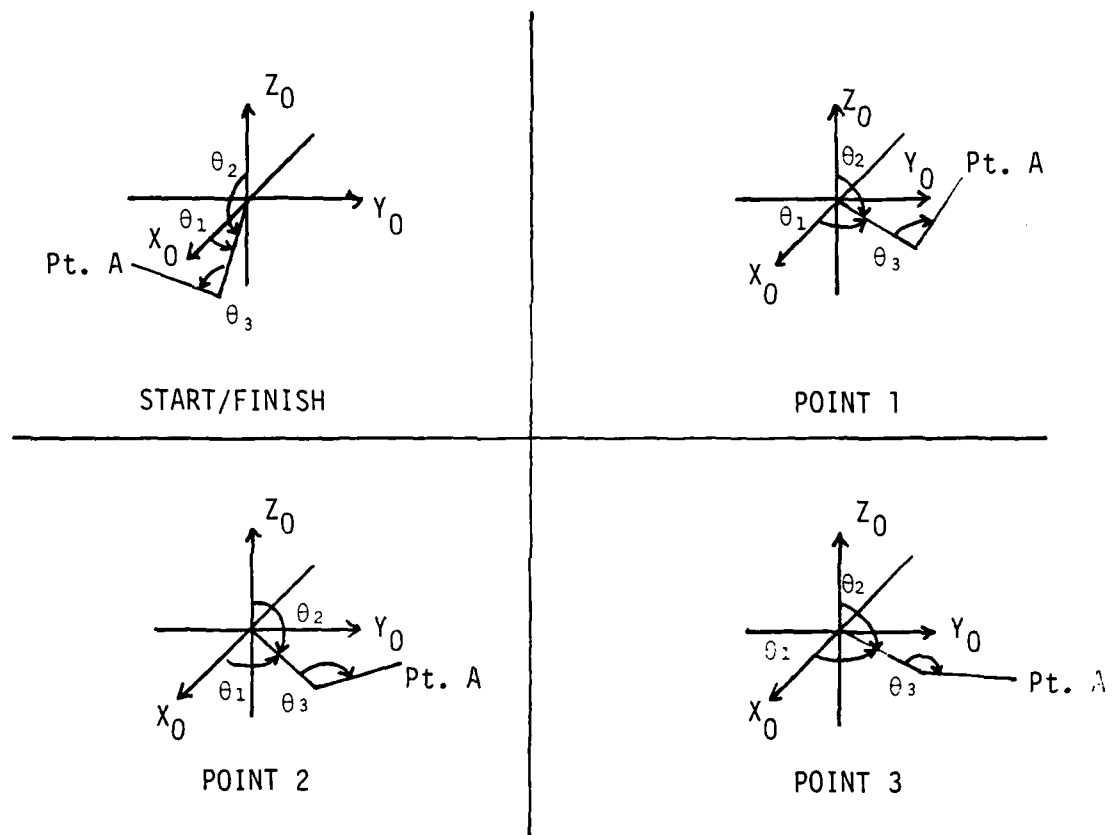


FIGURE 4.2 Arm Positions, Step 1, Implementation II

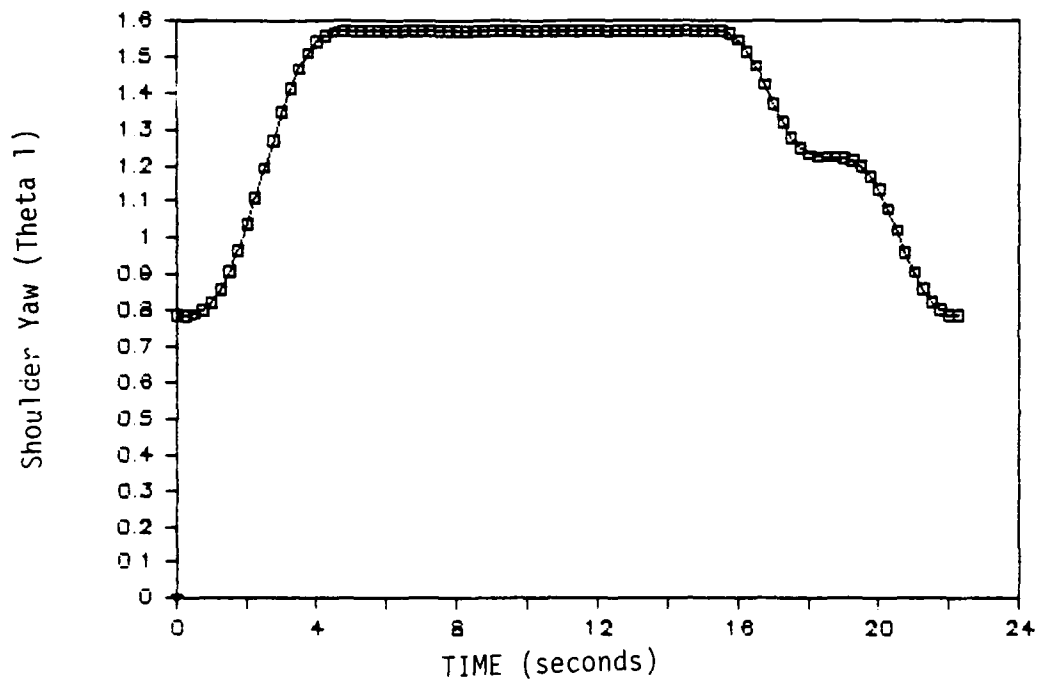


FIGURE 4.3 Shoulder Yaw Position, Step 1, Implementation II

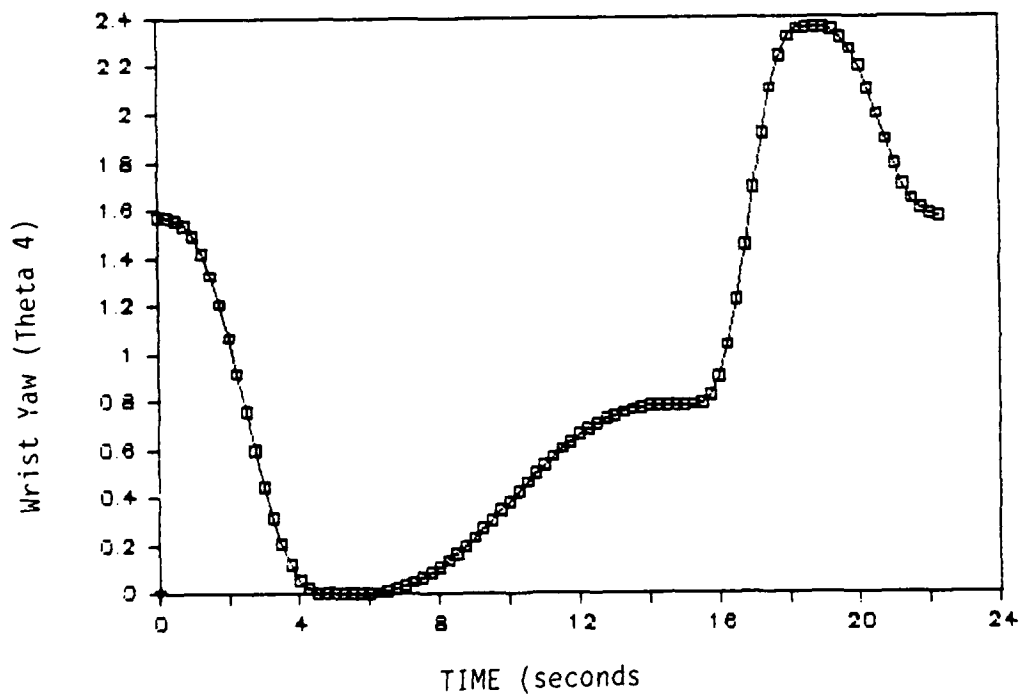


FIGURE 4.4 Wrist Yaw Position, Step 1, Implementation II

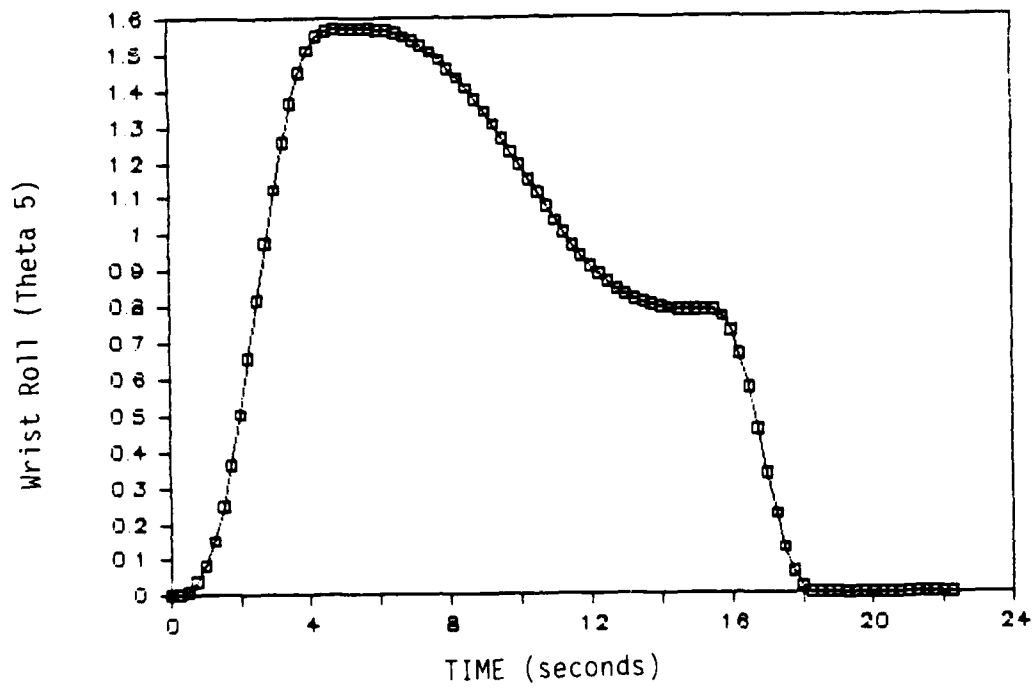


FIGURE 4.5 Wrist Roll Position, Step 1, Implementation II

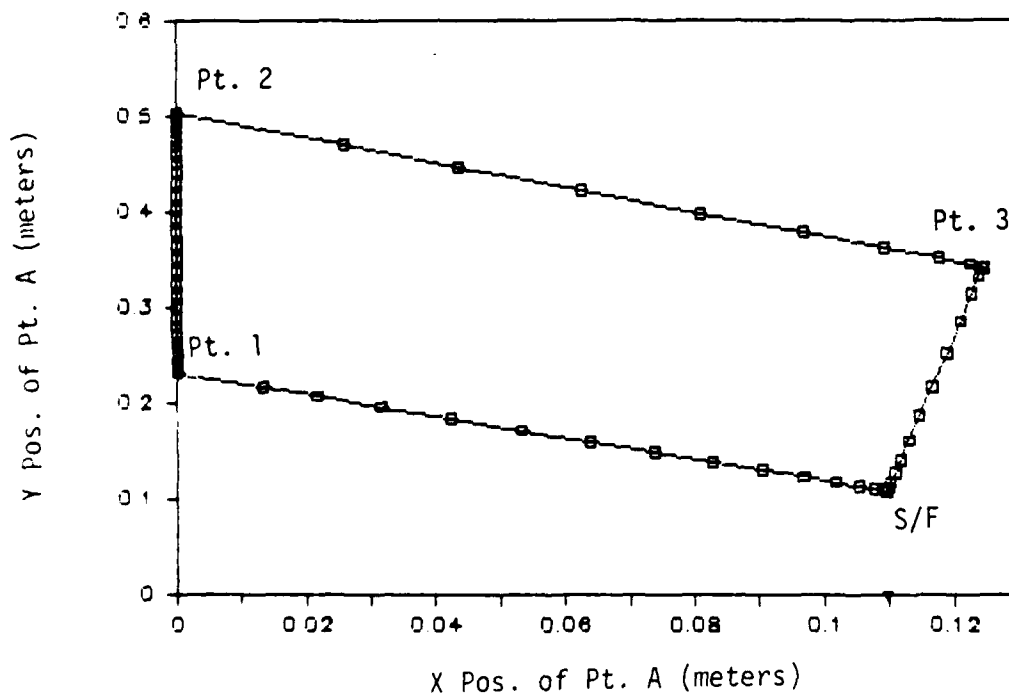


FIGURE 4.6 Y vs. X Position, Step 1, Implementation II

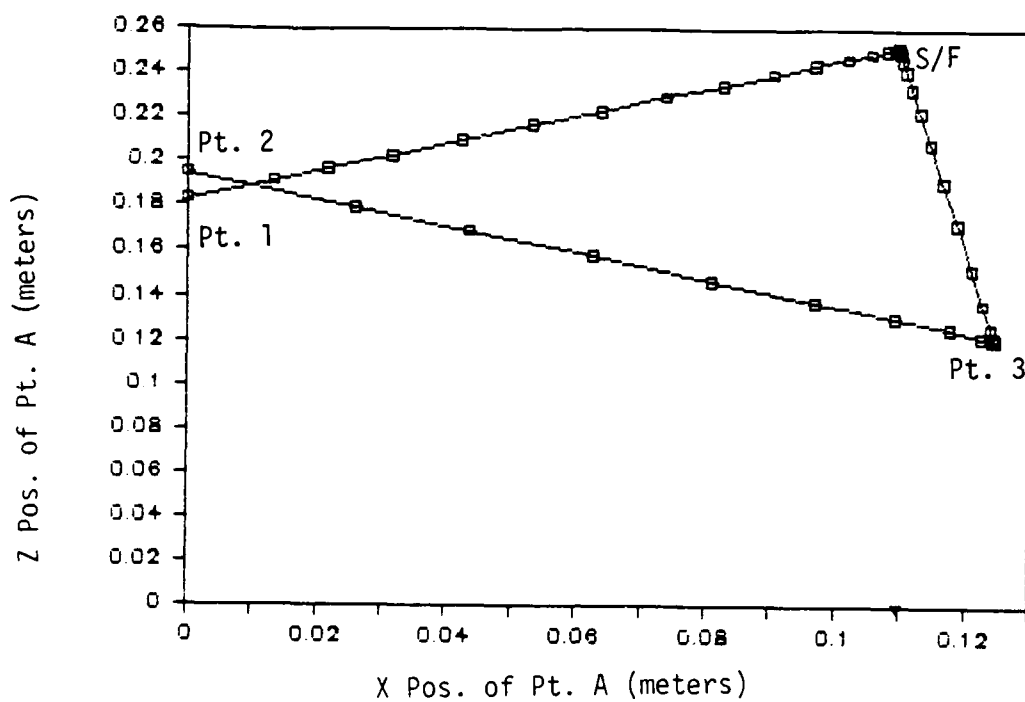


FIGURE 4.7 Z vs. X Position, Step 1, Implementation II

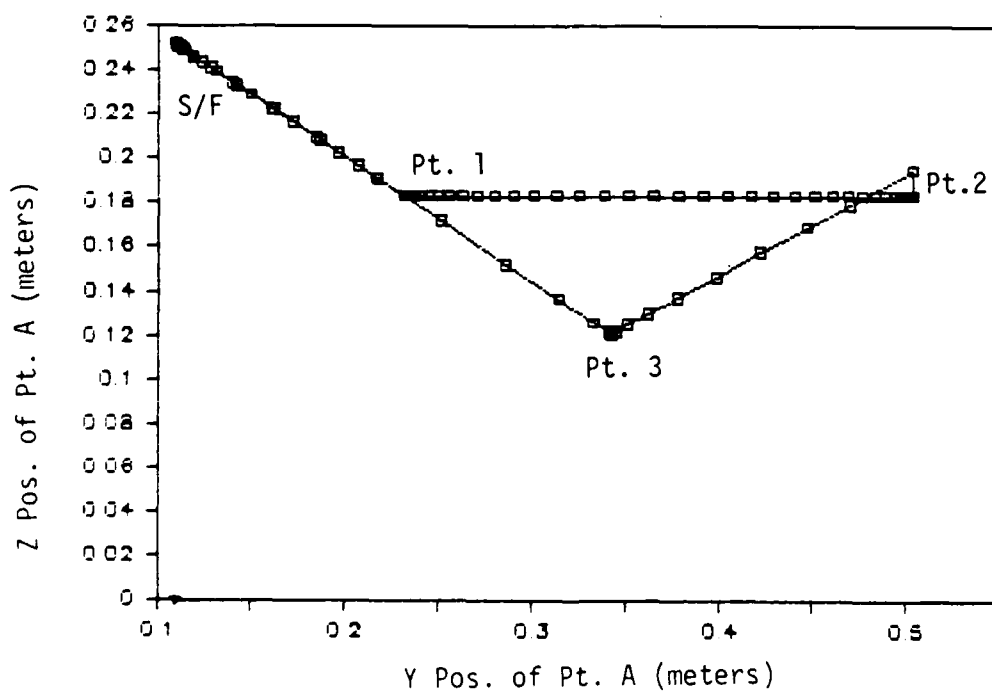


FIGURE 4.8 Z vs. Y Position, Step 1, Implementation II

The plots show the gradual acceleration and braking applied to the shoulder yaw, wrist yaw and wrist roll as the arm moves through all the points. The plots in X,Y,Z space show how the arm's end effector position (Point A) is kept on a straight line path through all the points. It also shows how the arm moves in a closed loop. It starts and finishes at the same point.

Step 2) Master arm entry, all values sent to a file.

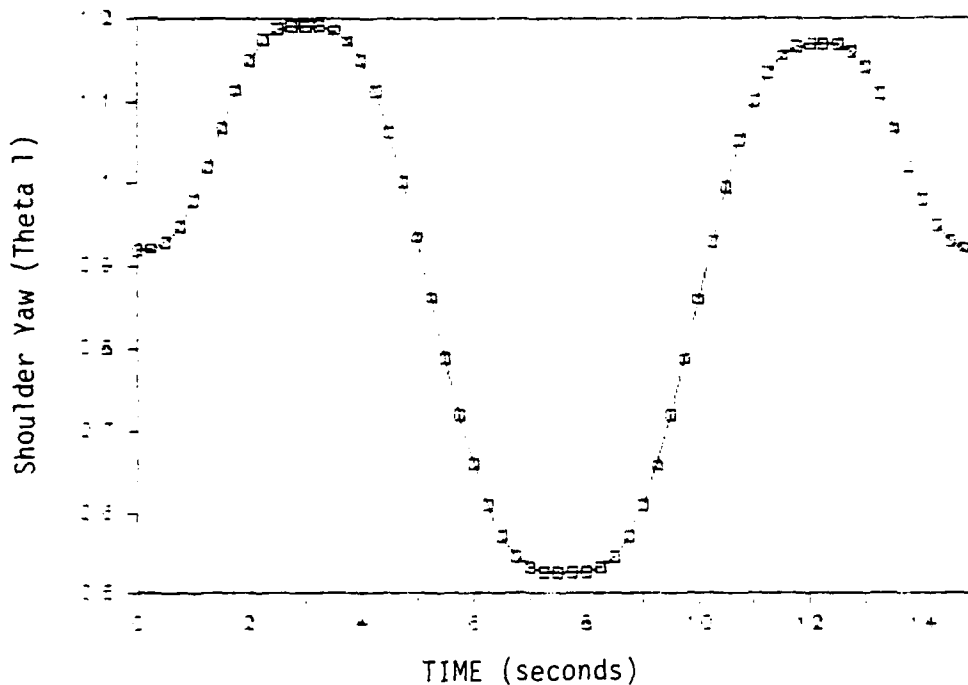


FIGURE 4.9 Shoulder Yaw Position, Step 2, Implementation II

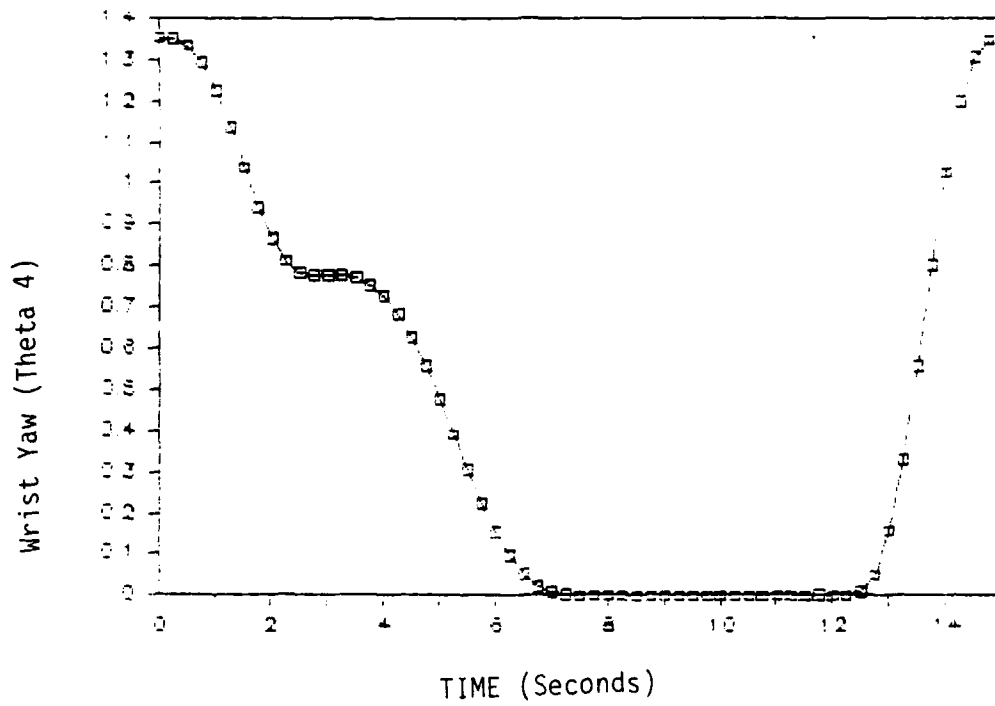


FIGURE 4.10 Wrist Yaw Position, Step 2, Implementation II

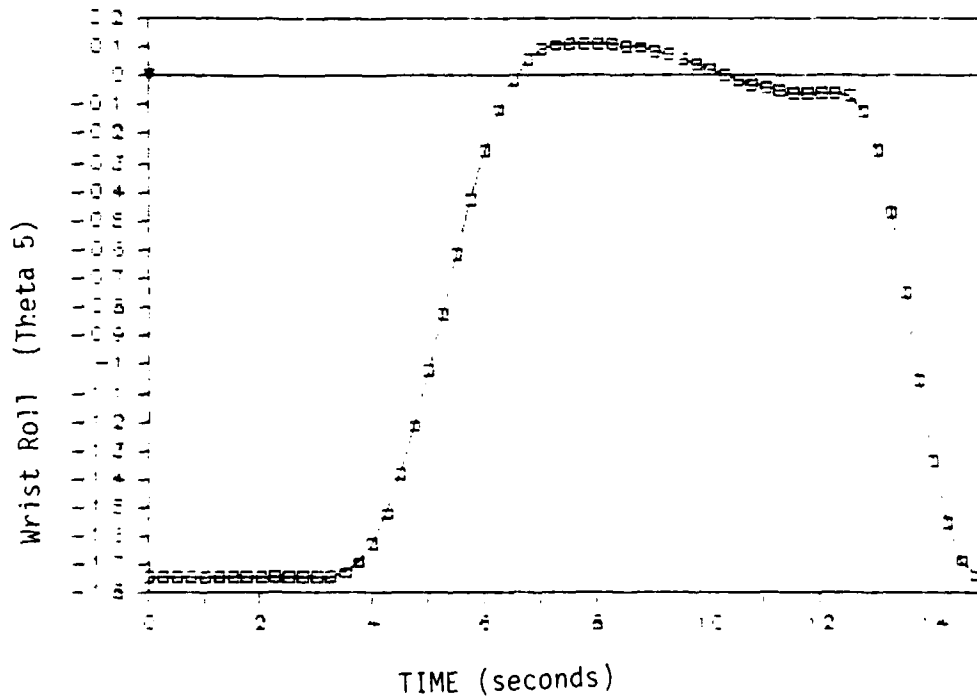


FIGURE 4.11 Wrist Roll Position, Step 2, Implementation II

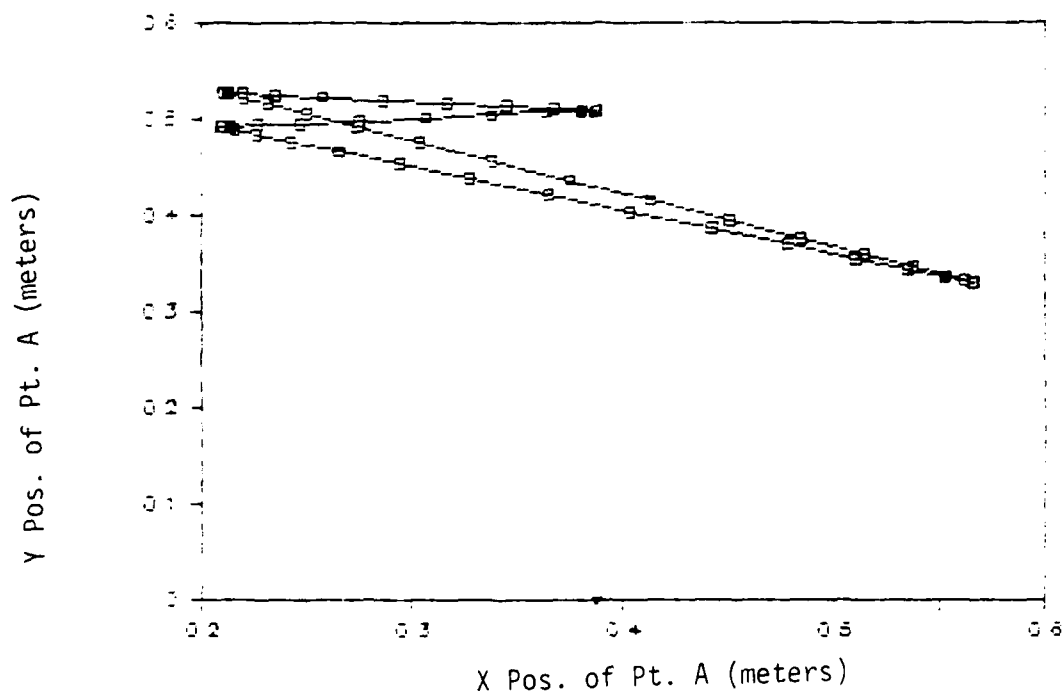


FIGURE 4.12 Y vs. X Position, Step 2, Implementation II

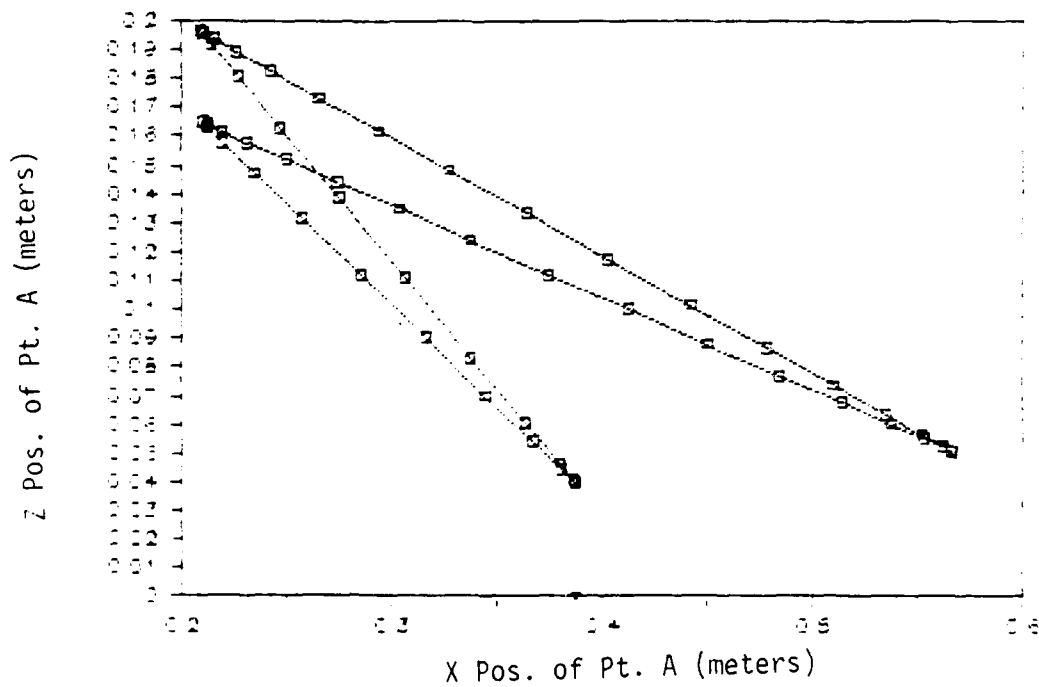


FIGURE 4.13 Z vs. X Position, Step 2, Implementation II

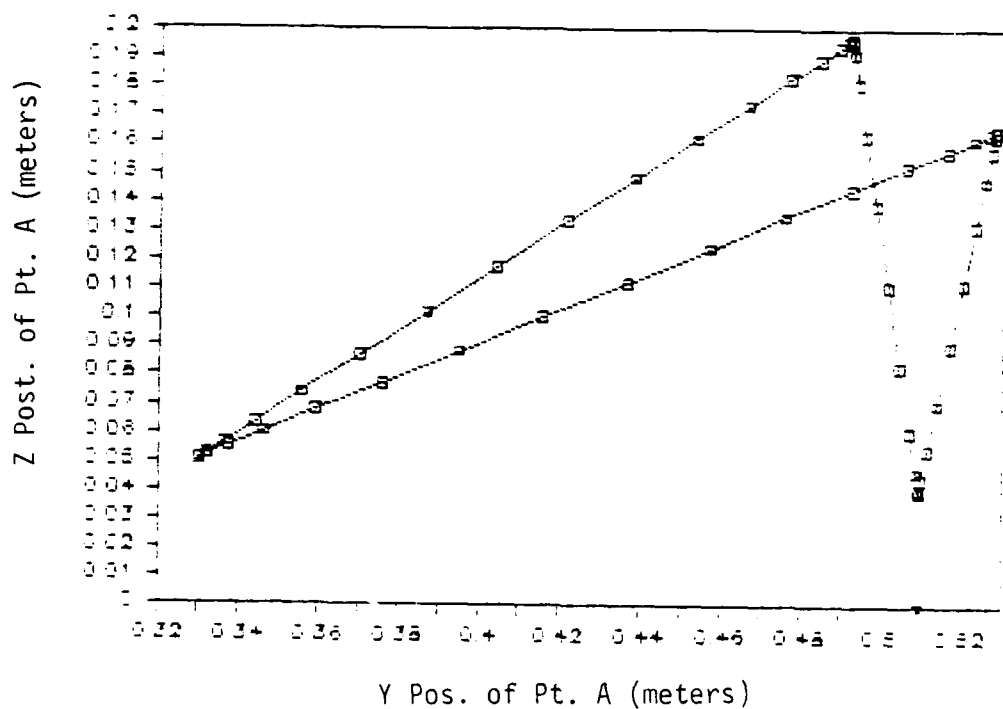


FIGURE 4.14 Z vs. Y Position, Step 2, Implementation I

Here again, the plots give the same type of results; the gradual acceleration and braking, the straight line paths and the closed loop trajectory.

The operator now has a great deal of supervisory control over the right arm. The only time the operator needs to use the master arm is to store points for a trajectory. He/she could then put the master arm in a set position and use the 'z' option to perform a movement task over and over. This will help free the operator's hands for other functions at the ICS. The extensions of this will be discussed in the next chapter.

CHAPTER 5

CONCLUSIONS

The two implementations discussed in Chapters 3 and 4 were incorporated into the existing BAT software. The operator can enter the supervisory control mode by either hitting the 'w' key or by pushing the black pushbutton between the joysticks on the ICS's bottom control panel. The code for the supervisory control mode is listed in Appendix B. While the BAT was in the laboratory (not in the neutral buoyancy environment), the software was tested. All of the BAT's right arm actuators were powered up except the shoulder yaw motor. It would place too much strain on the motor if it was driving the arm outside the neutral buoyancy environment. At the ICS the supervisory mode was selected and both implementations (unknown starting position to a known finish point; concatenating points) were run. Since the shoulder yaw motor was not powered up it could not be ascertained as to whether the path of point A (the end effector) was a straight line path. It was very apparent that the arm movement was not erratic and was very smooth. The joints that were powered did move with gradual acceleration and braking. When the second implementation was performed the arm did pass through all the points and returned to the finish point.

The supervisory control of the BAT was tested in the neutral buoyancy environment by the Space Systems Laboratory. While in master-slave control, one of the wrist motors was running intermittently. This caused a sudden jerk in the movement of the wrist since the two wrist

motors are coupled to give wrist yaw and roll. In the supervisory mode, when both implementations were tested, the same jerks also occurred in the wrist movement. Otherwise, the wrist moved to the points with a gradual acceleration and a gradual braking. The elbow moved smoothly throughout the test in both master-slave and supervisory mode. Shoulder yaw and pitch operated smoothly in master-slave control, but did not move at all when the supervisory options were selected. Due to an oversight in the development of the software, the shoulder pitch and yaw commands were not being sent to the control system. This was easily corrected but not in time to be tested again. Since only the elbow and wrist were being controlled, the end effector did not move in a straight line path.

Although the results of the test are inconclusive as to whether the supervisory control method will move the arm in a straight line path to a specified point, it does show that, in the neutral buoyancy environment, the elbow joint was controlled smoothly and gradually. The next test will be directed towards insuring all the joints are controlled properly. Upon completion of this step, the accuracy of the joint positions will be evaluated.

From the plots of the simulations in Chapter 2 and the plots of the implementations in Chapters 3 and 4, it is easy to see that for the given inputs the arm will move in a straight line path. With the test on the BAT in the laboratory, the results showed that the program flow was working properly and the arm was moving smoothly with gradual acceleration and braking. When the program is tested on the BAT again in the neutral

buoyancy environment, the only cause for error is if the relationship between the BAT's right arm joint angles and the joint angles which the central controller generates is not the same. This refers back to the assumption that the master arm's position gives the correct position of the BAT's right arm. This can be corrected by simply readjusting the integer/radian conversion between the master arm's analog to digital conversion and the angular position of the BAT's right arm joints. Once the integer/radian conversion is accurate, the previous results conclude that the arm's end effector will move in a straight line path smoothly while using this method of supervisory control.

With this method of controlling the BAT's right arm movements, many extensions can be incorporated into the BAT's software, so that the control relationship will be almost entirely supervisory. Other functions can be added to the software, so that the right arm can perform many other tasks by using keyboard entry at the central controller. For example, a function called rotate (arguments) can be developed. Here the orientation angles (wrist yaw and wrist roll) are changed according to the arguments of the function (i.e., rotate ($\pi/2$, 0) would change the wrist yaw $\pi/2$ radians and the wrist roll 0 radians). There could also be other functions like lift (arguments) where just the shoulder pitch and elbow angles would be changed, so that the arm's end effector would move in a straight vertical line. All these functions can be developed and stored in a library. The operator could then perform complete assembly tasks by simply typing in a list of these functions for the BAT to follow. For example, a list could be as follows:

small motor drive circuit can handle only 50 amps. This provides all eight drives (the camera tilt and pan, the five right arm motors and the left arm motor) necessary for the operation of the BAT. The control of the right arm is discussed in more detail in Appendix A, Section 3.

D) Propulsion Subsystem (see Figure A.8)

There are eight electric trolling motors pressurized for water-proofing. They combine the thrust to give the BAT six degrees-of-freedom. The main power subsystem supplies power to the power transistor modules which drive the motors. The modules contain pulse-width modulation drives for each of the eight motors. They are controlled from the control box propulsion controller through the relay box. The propulsion controller receives its commands from the ICS through the RRT. An inertial package provides attitude and rate feedback through the propulsion controller and the RRT to the ICS.

Aside from the video signals from the two cameras on the BAT, all of the BAT's input and output signals are passed through the RRT in the control box to the ICS. All the BAT's input signals originate at the ICS, and all the BAT's output is fed back to the ICS.

A.2 Hardware of the ICS

During the neutral buoyancy tests the BAT is controlled by the Integrated Control Station (ICS) (see Figure A.9). The ICS's frame is made of welded angle stock and mounted on wheels so it can be transported to test sites with ease. It has room to hold the operator, the controls and their displays, and video equipment.

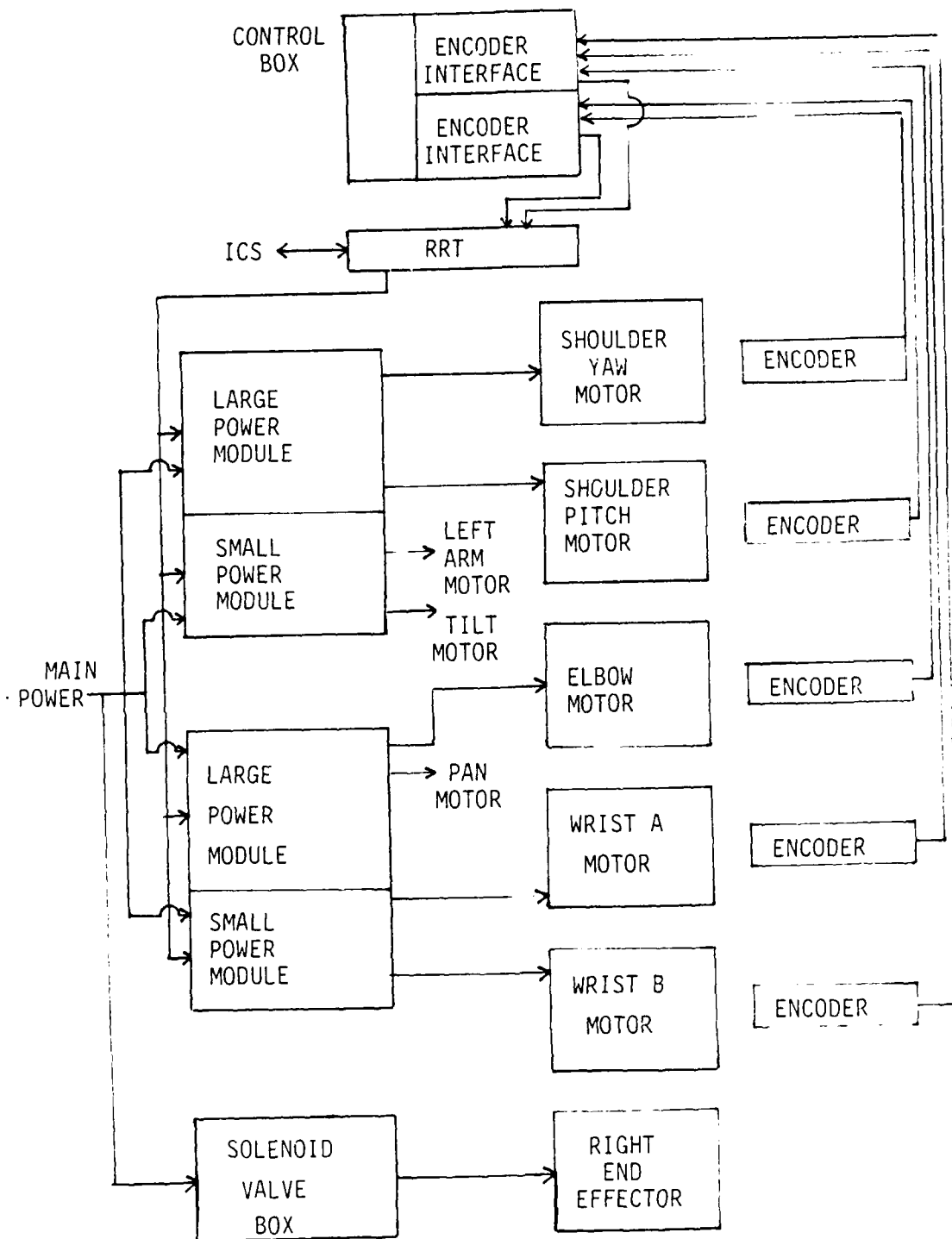


FIGURE A.7 Right Arm Subsystem

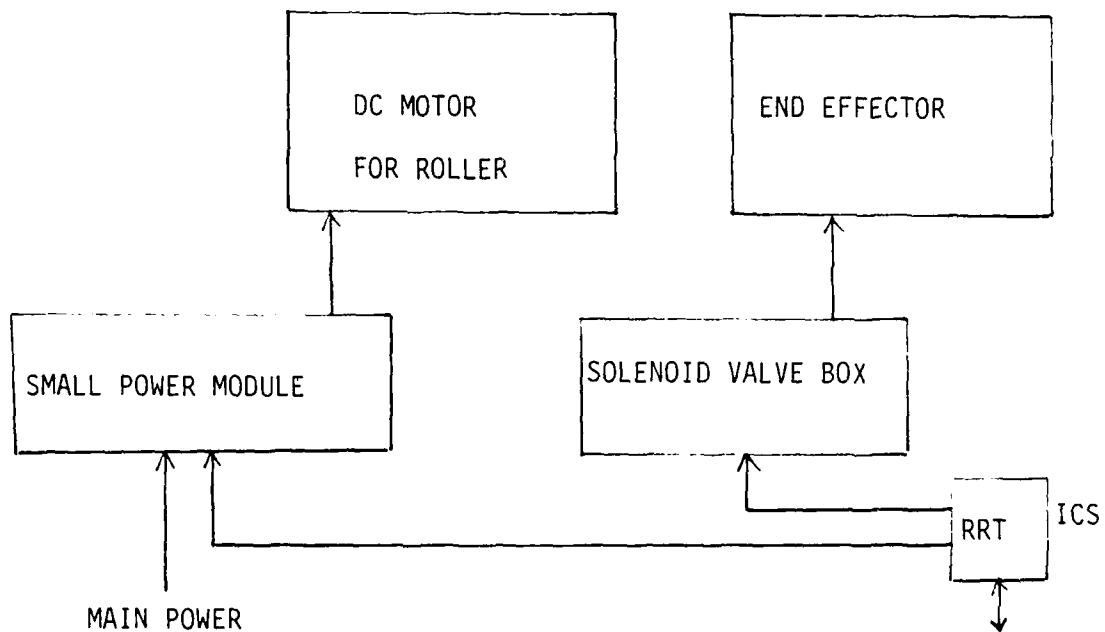


FIGURE A.6 Left Arm Subsystem

motors receive their input commands from the ICS through the RRT and two power modules. These power modules will be described in further detail in the right arm subsystem. Two optical encoders are used for tilt and pan position feedback to the interfaces in the control box. The cameras' video signals are connected to video hardware at the ICS through coax cables.

B) Left Arm Subsystem (see Figure A.6)

The left arm is a specialized actuator which can grasp a beam and move it laterally across the body of the BAT. It grasps the beam by means of a specialized end effector, which is opened and closed pneumatically. It is controlled by the ICS through the solenoid valve box and the RRT. At the base of the end effector is a roller. This roller can move a beam across the BAT. The roller is driven by a permanent magnet DC motor. The motor receives its power from the main power subsystem through a small power module (described in following section).

C) Right Arm Subsystem (see Figure A.7)

The right arm is a five degree-of-freedom manipulator with a grasping end effector. The end effector is controlled pneumatically from the ICS through the RRT. Each of the five degrees-of-freedom are driven by a permanent magnet DC motor. The motors are connected to the power modules, which are pulse-width modulation amplifiers. They are controlled from the ICS through the RRT. There are two power modules. Each power module contains four independent motor drive circuits; two large and two small. A large motor drive circuit can handle up to 100 amps, while a

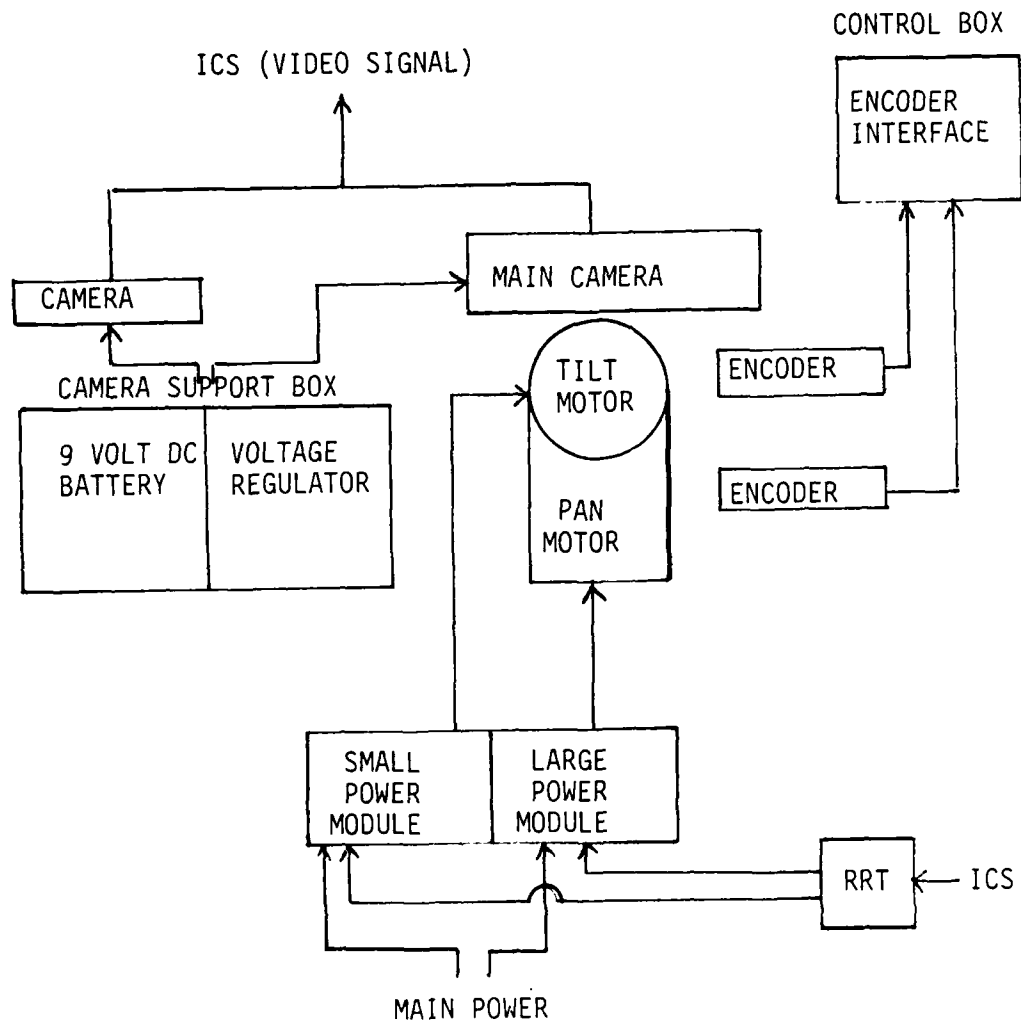


FIGURE A.5 Video Subsystem

supplies low pressure air to keep each box in the BAT watertight. The other tank supplies high pressure air, which is connected to the solenoid valve box. The solenoid valve box uses this high pressure to control three components:

- the left arm end effector
- the right arm end effector
- the main power relay.

Both end effectors receive their open/close commands from the ICS through the RRT. The Main Power Relay is an electronically-controlled pneumatically-operated switch which controls the power output of the main battery box. Its normal operating position is closed. It may be activated from the ICS or from a switch on the BAT. In the event of a communications failure or upon activation it opens, disconnecting the main power from the BAT's actuators and thrusters.

II. Action Subsystems

A) Video Subsystem (see Figure A.5)

The video subsystem has two commercial cameras, each housed in an aluminum tube with a front plexiglas plate. The main camera is mounted on a tilt and pan unit on the top of the BAT. A second camera is mounted above the right arm pivot and tracks the position of the end effector. The cameras are connected to a camera support box, which powers the cameras with two gel cell lead acid batteries hooked to a voltage regulator. The platform is a tilt and pan unit with two permanent magnet DC motors. The

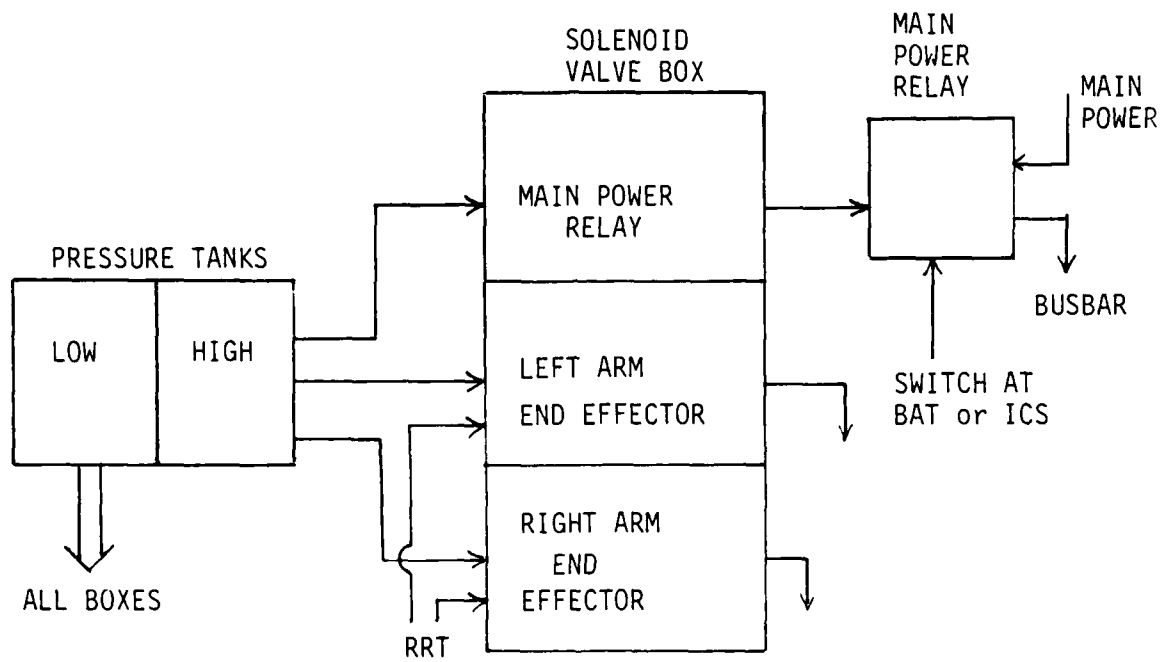


FIGURE A.4 Pneumatic Subsystem

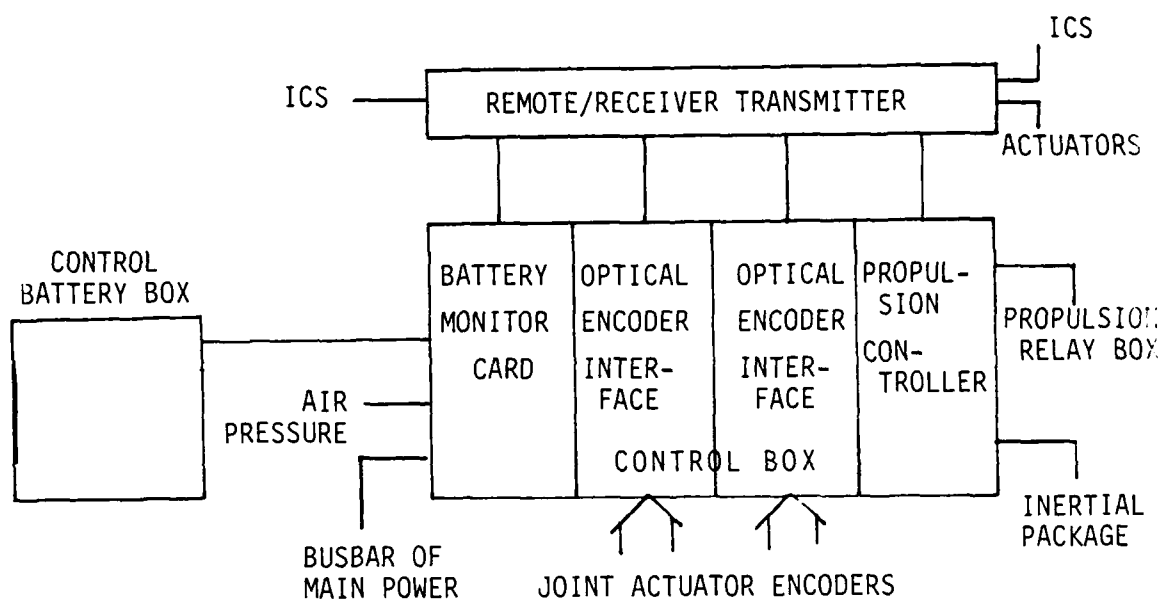


FIGURE A.3 Control Subsystem

- the power amplifiers of the right arm, left arm and the tilt and pan unit of the main camera;
- the relay box of the propulsion subsystem;
- the battery monitor card in the control subsystem.

B) Control Subsystem (see Figure A.3)

The control subsystem has two components; the control battery box and the control box. The control battery box consists of 12 Vdc, 5 Vdc and ± 6 Vdc power supplies which supply the power for the solenoid valves, the control logic and the communications link, respectively. Inside the control box is a battery monitor card, two encoder interfaces, a propulsion controller and communications hardware. The battery monitor card samples and digitizes the voltage levels of the main batteries and the air pressure transducer. The two optical encoder interfaces are used with each actuator in the action subsystem. They convert each encoder's output and transmit the outputs to the remote receiver/transmitter (RRT). The propulsion controller receives information from the inertial package of the propulsion subsystem. It also sends commands from the RRT to the propulsion subsystem. The control box communicates with the ICS through the telemetry uplink and downlink. This is implemented by the RRT which combines the uplink multiplexer with the downlink digital demultiplexer. It also contains the analog transceiver.

C) Pneumatic Subsystem (see Figure A.4)

The pneumatic subsystem consists of two separate air tanks. One

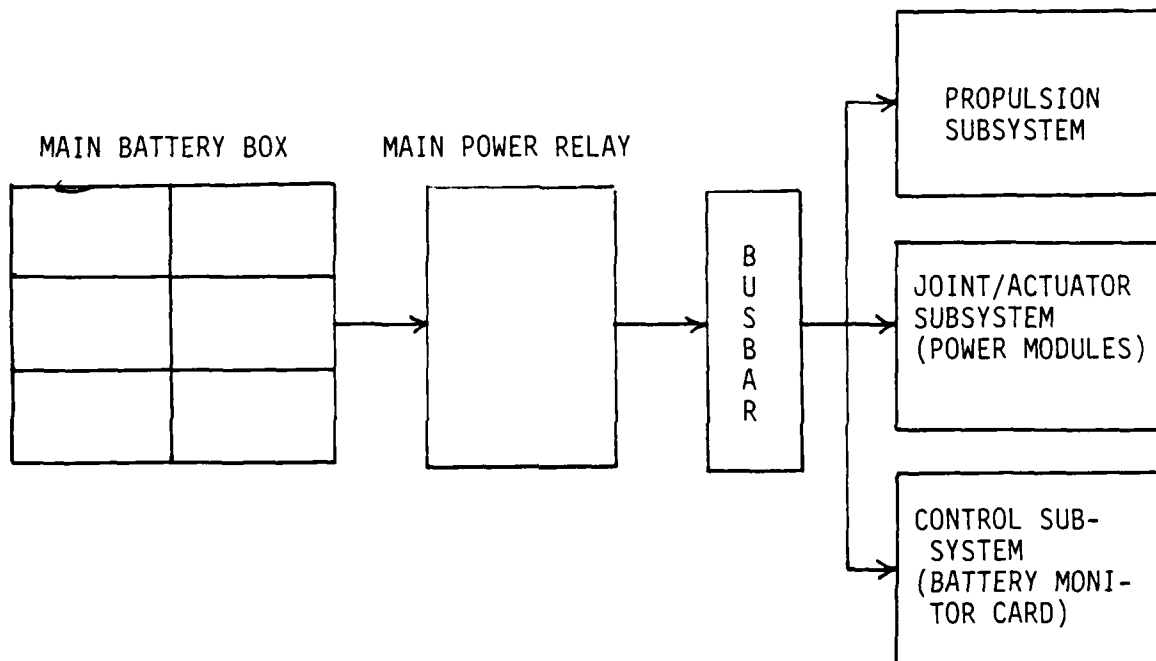


FIGURE A.2 Main Power Subsystem

The Beam Assembly Teleoperator (BAT) (Figure A.1) is a self-contained free-flying teleoperator. It was developed to be used under water in neutral buoyancy structural assembly tests. The BAT's flotation panels provide most of its neutral buoyancy. It is controlled from an above-water control station, the ICS (described in Section A.3). The BAT can be broken down into four support subsystems and three action subsystems.

I SUPPORT

- A) Main Power Subsystem
- B) Control Subsystem
- C) Pneumatic Subsystem
- D) Propulsion Subsystem

II ACTION

- A) Video Subsystem
- B) Left Arm Subsystem
- C) Right Arm Subsystem

I. Support Subsystems

- A) Main Power Subsystem (see Figure A.2).

The main battery box contains six battery packs connected in parallel. Each battery pack contains three gelled electrolyte lead acid cell batteries connected in series. Eighteen volts at 20 amp-hours is supplied by each battery pack. The use of the on-board battery box eliminates the need for a cumbersome umbilical and cuts down on power loss through transmission. The main battery box is connected through the main power relay (described in the pneumatic subsystem) to the bus-bar. The distribution bus-bar connects the main battery box to:

APPENDIX A
DESCRIPTION OF THE BAT

The Beam Assembly Teleoperator was developed and is used by the Space Systems Laboratory at M.I.T. to experiment with the man-machine interface for assembly tasks. This appendix describes the system in detail. The appendix is broken down into three sections:

- A.1 Description of the BAT
- A.2 Hardware of the Integrated Control Stations (ICS)
- A.3 Software of the ICS.

A.1 Description of the BAT (see Figure A.1)

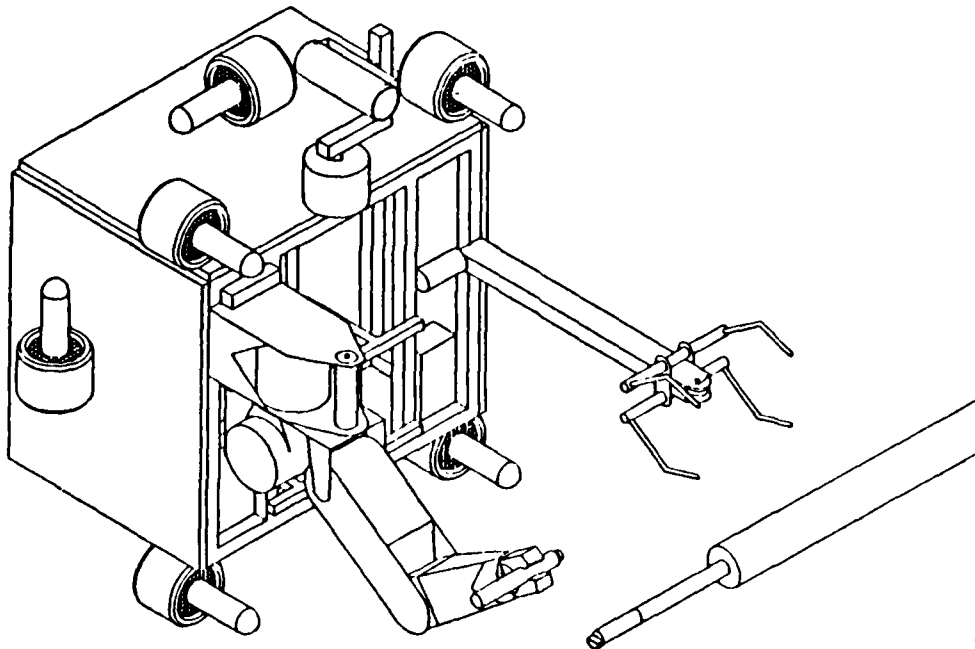


FIGURE A.1 Sketch of the BAT

REFERENCES

- [1] Robot Technology: Volume I - Modelling and Control, Philippe Coiffet, Prentice-Hall, Englewood Cliffs, N.J., 1983.
- [2] Mechanism Design: Analysis and Synthesis - Volume I, Arthur G. Erdman and George N. Sandor, Prentice Hall, Englewood Cliffs, N.J.,
- [3] "Design and Control of a Beam Assembly Teleoperator", Eric B. Shain, S.M. Thesis, Dept. of Aeronautics and Astronautics, M.I.T., May 1983.
- [4] "Results of the MIT Beam Assembly Teleoperator and Integrated Control Station", John R. Spofford and David L. Akin, Space Systems Laboratory, M.I.T.

- move(arguments)
- rotate(arguments)
- lift(arguments)
- move(arguments)

This list of functions can be stored as a macro, so that the operator can simply hit a function key (F1,F2, etc.) or type in the macro's name and the right arm of the BAT would perform the listed tasks. The operator can construct these lists, so that the arm would be performing a complete assembly task. All these functions (except the end effector's grasp() and release() functions) can be developed by simply modifying the supervisory method developed in this thesis.

The use of supervisory control on the BAT's right arm would free the operator's hands from the master arm and let the operator concentrate on other duties at the ICS. Supervisory control will increase the capabilities of the BAT and help improve the interface between man and machine during assembly operations.

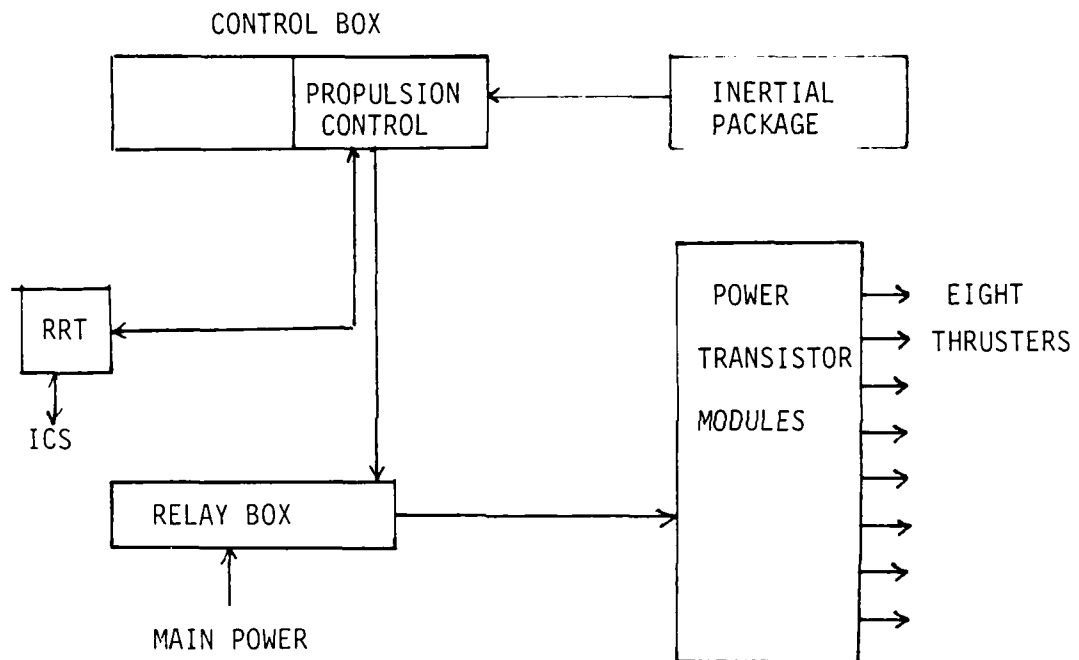


FIGURE A.8 Propulsion Subsystem

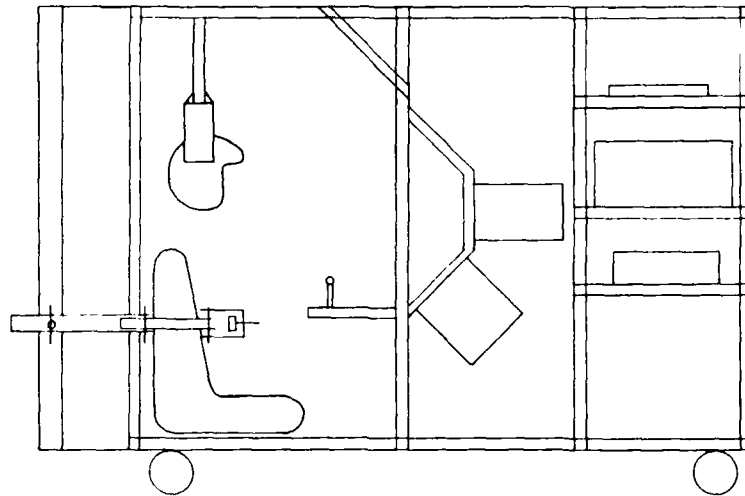


FIGURE A.9 Sketch of the ICS

The ICS consists of five subsystems:

- A) ICS Controls and Displays
- B) Central Controller
- C) Joint/Actuator Control System
- D) Propulsion Control System
- E) Communication Link.

A) ICS Control and Displays

The ICS controls and displays system takes up most of the area of the ICS. It contains the helmet for video control, the master arm and the control/display panels. The helmet is located above the operator and is

mounted on a gimbal transducer system. This system converts head movements into tilt and pan commands used by the central controller to control the BAT's main video camera. This helmet may be swung out of the way and back-up camera controls on the control/display panels used instead. The master is a five degree-of-freedom kinematic duplicate of the right arm on the BAT. The operator straps his/her arm on the master arm and positions it to where he/she would like the BAT's right arm to go. The positions of each joint of the master arm are read by the central controller, and transmitted through the joint/actuator control system to the BAT's right arm. Both the master arm and helmet systems implement master-slave control for the right arm and video subsystems, respectively.

There are four control/display panels (see Figure A.10) situated in front of the operator. The functions of all the controls on these panels can be changed through the software of the central controller (described in the next section). The top panel contains back-up controls for the propulsion control of the BAT and the back-up joystick control for the tilt and pan of the video subsystem. The second panel from the top contains 5 knobs which can operate in two modes. In one mode, the knobs can tune the gains and positions of the joint/actuator control system. In the other mode, the knobs can control the five joints of the BAT's right arm. The modes can be switched using the software in the central controller. There is also a back-up switch that can control the opening and closing the the end effector of the BAT's right arm. A lever is situated on the right of the panel to control the motor for the left arm roller. The third panel

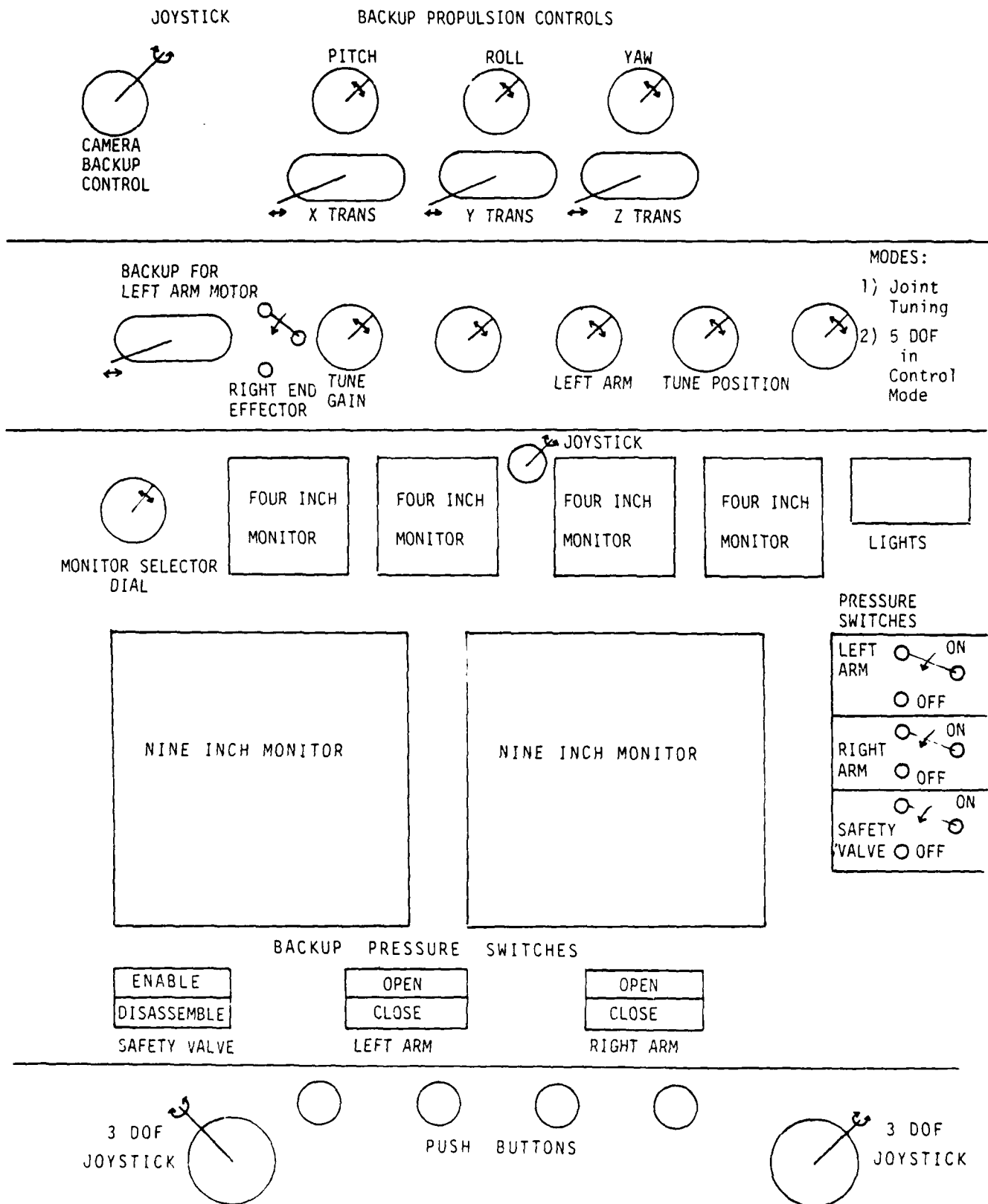


FIGURE A.10 ICS Control Panels

from the top is the largest and contains the video monitors for the ICS. There are 2 nine inch monitors and 4 four inch monitors above them. On the panel is a dial which is used to switch the display around on the different monitors. These monitors can display images sent from the cameras on the BAT or graphics produced by the software of the central controller. The graphics display is used to show the system settings and the current status. Other cameras, such as payload bay cameras, may also be hooked into the system and shown on these monitors. Before being connected to the monitors, the cameras are hooked into video equipment located on a rack behind the panels. Above the monitors on the right side is a set of lights which monitor the duty cycle for each actuator on the BAT. To the right of the panel are the pressure switches which control the high pressure of the pneumatic subsystem of the BAT. The switches control the right arm end effector, the left arm end effector and the safety valve. They go directly to the communications link and bypass the central controller. Below the video monitors are three backup buttons for the three pressure switches. The bottom panel contains the primary controls for the BAT's propulsion subsystem. These two joysticks (each joystick has three degrees-of-freedom) are used to control the BAT's movements. These controls can also be changed by software to control the BAT's right arm. Four pushbuttons are located between these joysticks. These buttons are used as backups for selecting options in the software (these options are described in the next section). All the outputs from the master arm, the helmet gimbal system and the panel controls (with the exception of the pressure controls) go to an analog-to-digital converter for processing. The analog-to-digital converter digitizes

and multiplexes these inputs for use by the central controller software.

B) Central Controller

The central controller is located on a rack behind the ICS control/display panels. The central controller is an IBM Personal Computer with 256K RAM, two disk drives, a math 8087 coprocessor and special interface boards to communicate to the other subsystems (see Figure A.11). The central controller:

- interfaces with the controls and displays of the ICS the the analog-to-digital converter.
- commands and monitors the joint/actuator and propulsion control subsystem.
- provides integrated control functions.
- records test data.

System status (such as battery voltage levels) is sent from the control box on the BAT through the communications uplink to the central controller. The software which controls the BAT system is described in detail in Section A.4.

C) Joint/Actuator Control System

The joint/actuator control subsystem consists of two parts; a monitor/buffer card and the joint/actuator control cards (see Figure A.12). The monitor/buffer card is a standardized Z-80 microprocessor card which buffers the commands from the central controller and retransmits them to the joint/actuator control cards. It interfaces the central controller with the joint/actuator cards. There are four joint actuator cards, each of which uses a Z-80 microprocessor. Each card implements closed-loop

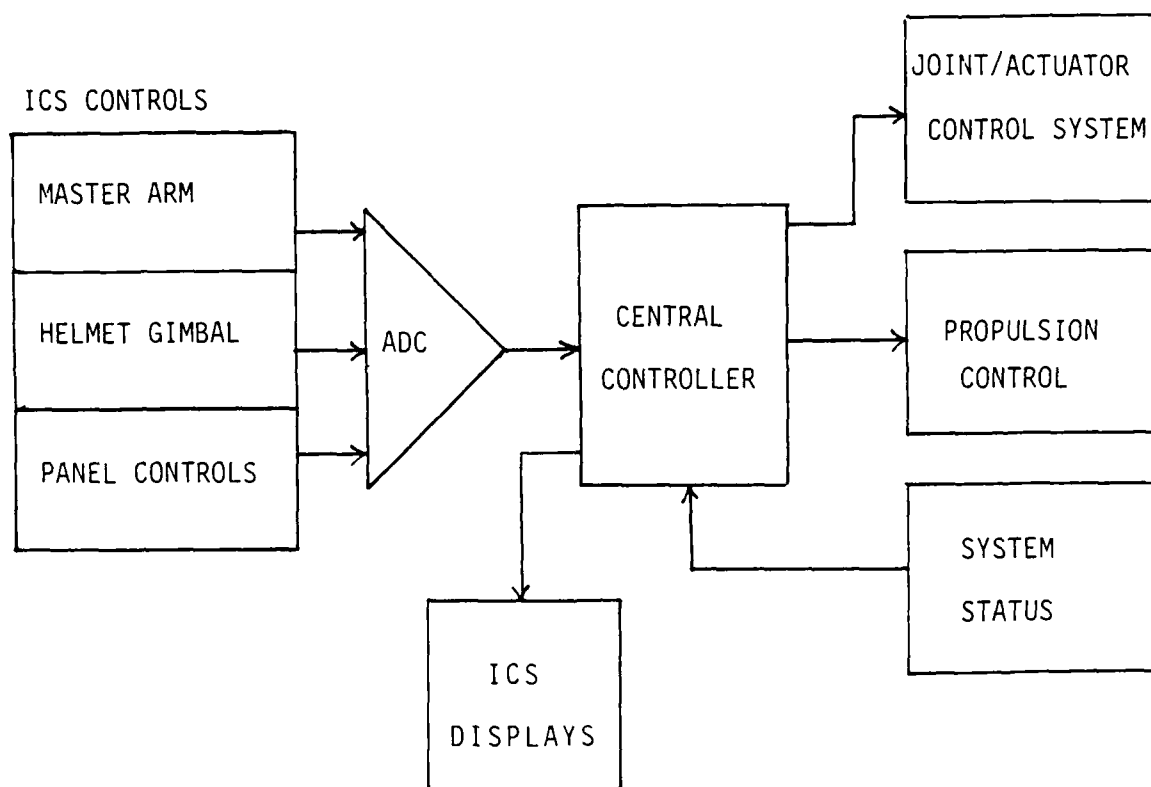


FIGURE A.11 Central Controller

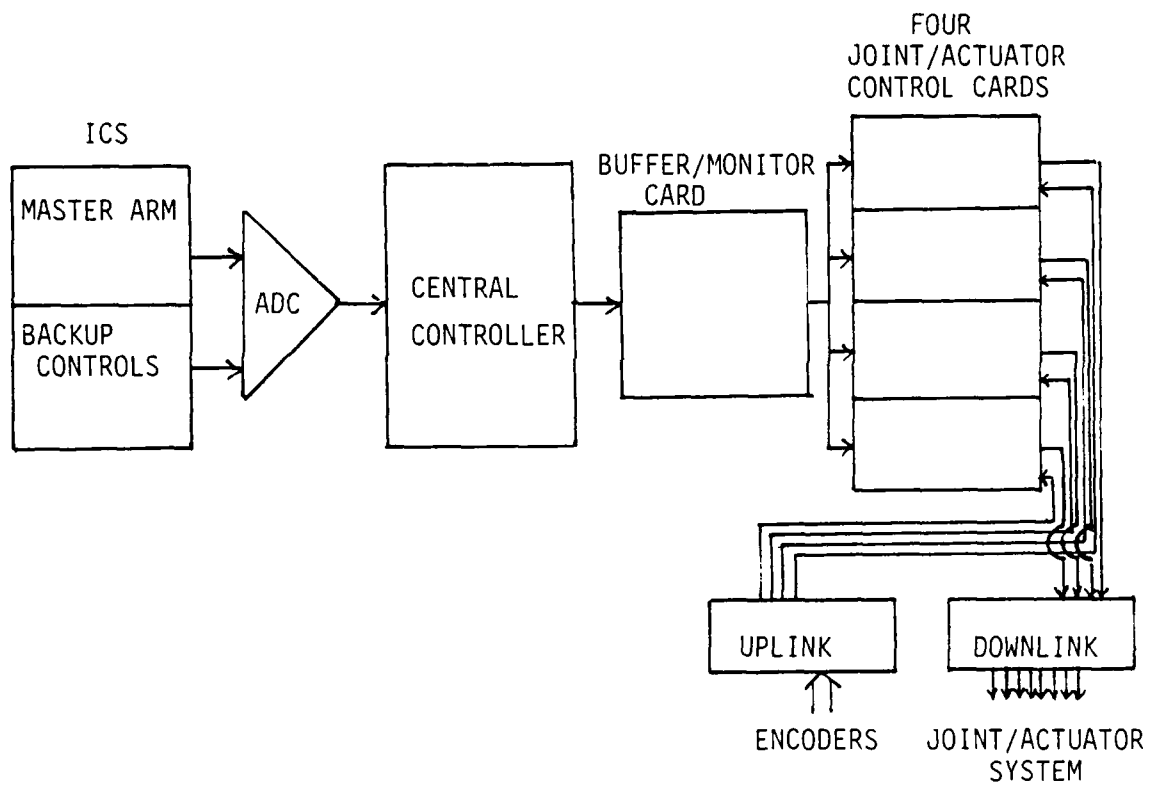


FIGURE A.12 Joint/Actuator Control System

control for two of the BAT's eight actuators. Control algorithms and parameters are downloaded to the cards from the central controller. They generate PWM drive signals and transmit them through the communications downlink to the BAT's five joint actuators of the right arm and the actuators for the left arm roller and for the camera tilt and pan. Each of the eight actuator positions are transmitted through the communications uplink back to the cards.

D) Propulsion Control System

The propulsion controller is implemented with a microprocessor card (see Figure A.13). The propulsion control card:

- accepts rotation and translation commands from the central controller
- performs the transformations necessary to convert rotation and translation commands to individual thruster commands.
- generates PWM drive signals for the power modules of the Propulsion subsystem.
- receives feedback from the BAT's inertial package through the communications uplink.

E) Communication Link

The communication link is a two megabits per second link which connects the control system to the BAT (see Figure A.14). The communication link consists of three parts; the downlink digital multiplexer, the uplink digital demultiplexer and the analog transmitter and receiver. The BAT has a similar link on-board. The speed of the communications link can

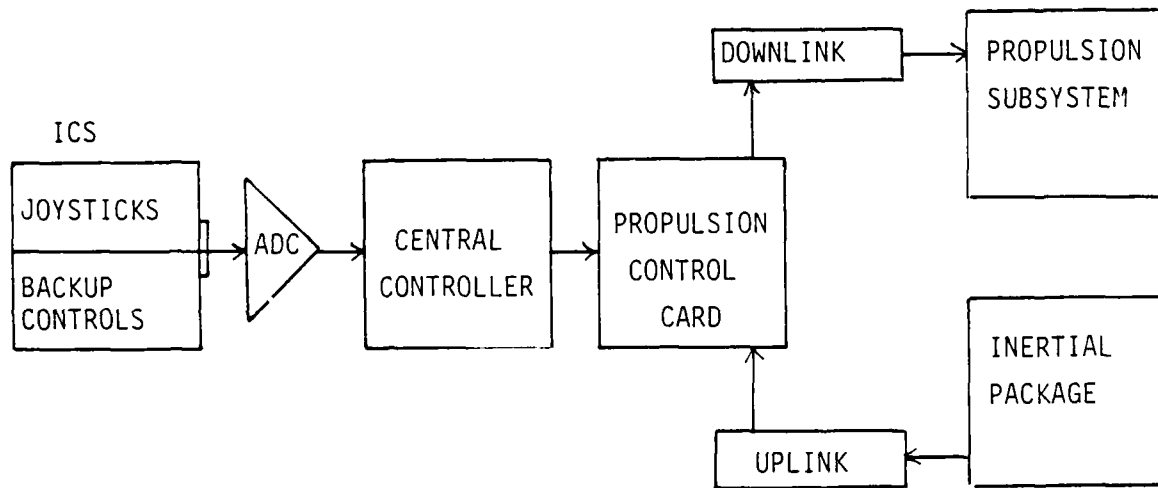


FIGURE A.13 Propulsion Control System

be upgraded if necessary. The digital multiplexer and demultiplexer are on separate cards in the control station whereas they are on the one card in the BAT. The analog receiver and transmitter are combined into a transceiver.

The hardware of the ICS and the BAT combine to make an effective teleoperator system. The interface between the operator and the system is in the software. The software can be used to change much of the way this interface occurs.

A.3 Software of the ICS

The central controller interfaces the ICS controls with the action and support subsystems of the BAT. The vast majority of the central controller software is written in the C programming language. The rest of the software is in assembly language. The modular aspect of C makes the software easier to follow and debug. Including the MAIN() function, the software is composed of 34 functions and the standard functions included in the C compiler. The 34 functions are all contained in three files; BAT1.C, SUPPORT1.C and SUPPORT2.C. These three files are linked together to make up the operating program for the BAT system.

The three files contain all the software to start up, run, adjust and shut-down the BAT system. It also displays all the necessary information on a monitor at the ICS. The file BAT1.C contains the MAIN() function which calls everything else (see Figure A.15).

BAT1.C calls:

- A) initializebat()
- B) calibratebat()

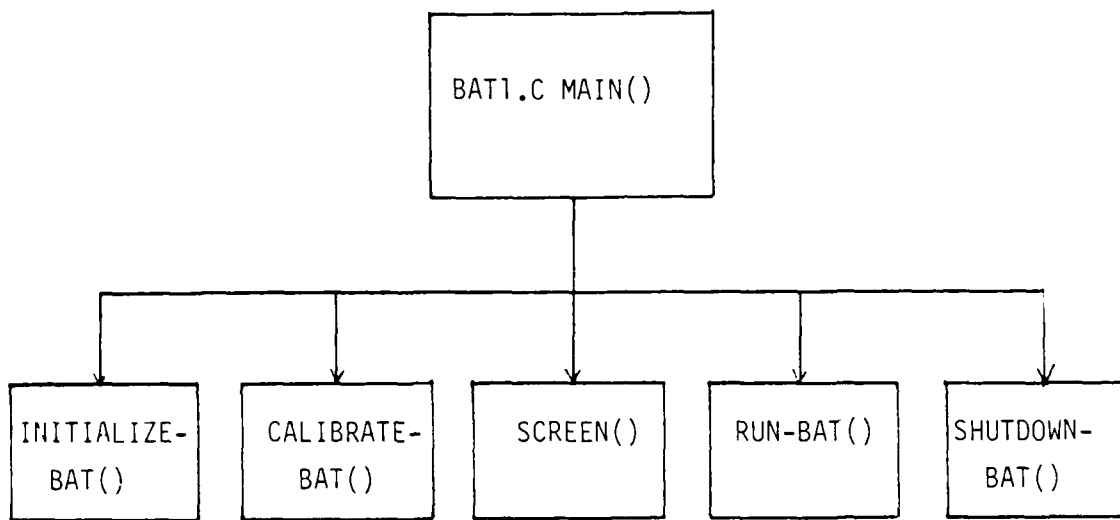


FIGURE A.15 Main() Function

- C) screen() and screen routines
- D) runbat()
- E) shutdownbat()

SUPPORT1.C and SUPPORT2.C contain functions which are called from these five main functions.

A) Initializebat() (see Figure A.16)

This function configures the ICS interface and initializes the joint/actuator and propulsion control subsystem. It first calls batports() which configures the ports on the interface board in the central controller. Batports() does this by calling setport() for each of the five Intel 8255 Programmable Parallel Interface chips on the board. Setport() sets an 8255's port to the input or output mode. It calls output() to send out the address and the control word to the chip. Output() and input() are functions which are written in Assembly language for speed and both do precisely what their names imply. They output information to a desired port from the central controller, and input information from a desired port to the central controller. After the ports are configured, initializebat() calls output() to initialize the joint/actuator, video and propulsion control systems. It does this by either outputting zero to the port or strobing the port. The main power relay is then checked by input() to see if the relay has been activated. If the main power relay is activated, the operator is notified and must deactivate the switch. The initialization process is then complete and control is returned to MAIN().

B) Calibratebat() (see Figure A.17)

This function calibrates the two 3 degrees-of-freedom joysticks used for primary control of the Propulsion subsystem. It informs the

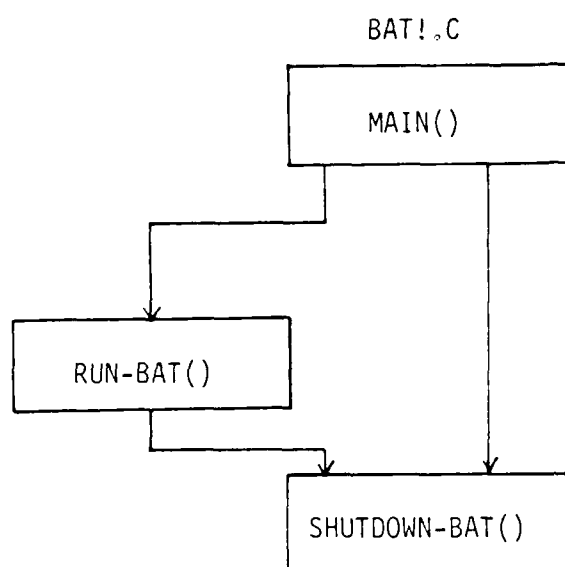


FIGURE A.20 Shutdown BAT() Function

E) Shutdownbat() (see Figure A.20)

This function simply shuts down the BAT system by strobing the buffer/monitor and propulsion controller cards. It then returns to the MAIN() function. The MAIN() function then calls a screen routine and finishes.

actuator cards from a file. The screen() function is then called to restore the original display. Control is then returned to the beginning of the main loop.

i) 'i' - calls adjustpots(). This function uses the convert() and adjust() functions to adjust the potentiometers used for control inputs on the ICS. It can change the minimum or maximum limits, the of sets or the interface between potentiometers and the ICS. It then calls the screen() function to restore the original display and returns control to the beginning of the main loop.

j) 'b' - this option is still being developed. It calls many functions such as testroutine(), readmast(), readjm(). Its purpose is to test the feedback of the joint/actuator control system. It attempts to bring the feedback to the central controller. Currently the feedback stops at the joint/actuator cards. It then restores the display using screen() and returns control to the beginning of the main loop.

k) '1,2,3,4' - these four options call shut cardoff(). This function shuts down the card which is selected. It sets the card's values to zero and calls transmast(). By shutting down the card, it also turns off that joint/actuator. It then returns control to the beginning of the main loop.

6) After these options are selected (or not selected as in the first case), synchronization is tested and the loop begins again. It does not exit the loop until carriage return is selected.

the propulsion card are reset using `outport()`. Initial values for address, data and command are sent to the buffer/monitor card via `transmast()`. Control is then returned to the beginning of the main loop.

f) 'c' - changes the camera control system. This is done by switching the value of the variable "headcontroller" (which was initialized as true). If the variable is true, the helmet gimbal system controls the video camera on the BAT. If it is false, the backup joystick on the ICS would then control the camera (Chapter 3, Figure 3.2). Control is then returned to the beginning of the main loop.

g) 'p' - switches the value of "puma". This variable determines whether the propulsion system is on or off. Control is then returned to the beginning of the main loop.

h) 't' - calls `setupjoints()`. `Setupjoints()` clears the screen, puts up a menu and retrieves a selection from the operator. The operator determines how he/she would like the joint control parameters set up. It gives the operator the capability of changing the gain of the controller, parameters of the controller and the type of control algorithm on the joint/actuator card (Proportional, Proportional-Derivative, State Feedback). It does this for each of the joint/actuators in the system by calling `tunejoint()`. Once the operator is satisfied with the values selected for each joint, the function `setupjoints()` writes the values to a file. The operator can also select the `loadfromfile()` function. This function loads the values (determined at some other time) of the joint/

putted from the keyboard and returns its value. If no character has been inputted, it returns zero. A menu is printed on the screen to help the operator select an option. The different options are outlined below.

a) No input - if there is no input (i.e., the operator does nothing), the central controller calls `inport()` to check the status of five switches. These switches are located between the two joysticks on the bottom panel (see Chapter 3, Figure 3.2). Currently, since these switches have no function, the central controller simply checks if the switch settings were changed. If the settings were changed, the central controller prints a number corresponding to the switch settings and returns to the beginning of the main loop (checking synchronization). If the switches were not changed it just returns to the beginning of the main loop.

b) Carriage return (value equals 13) - breaks out of the main loop and goes to `shutdownbat()`. It shuts down the system.

c) 's' - switches the value of "joyflag". Switching is defined as whatever the variable was, it is now the opposite (i.e., if the variable was true it is changed to false). It controls whether the joysticks are used to control the propulsion subsystem or the BAT's right arm. Control is then returned to the beginning of the main loop.

d) 'd' - switches the value of "draw". This determines whether or not the operator wants the actuator and thruster levels displayed on the screen. Control is then returned to the beginning of the main loop.

e) 'r' - resets the control system. It reinitializes the system as it did prior to starting the main loop. The buffer/monitor card and

played on the screen using the values stored by either the master arm or the joysticks. If "draw" is false (the default) then this step is skipped. The values stored are then sent to the buffer/monitor card by transmast().

4) The propulsion subsystem then follows a similar procedure. If "joyflag" is true, the convert() function reads the analog inputs of the joysticks. The joystick offsets are then added and the values are adjusted. The values are stored in an array. If "joyflag" is false then the convert() function reads the backup controls on the ICS. After these values are adjusted they are stored in an array. As before, the variable "draw" is checked. If "draw" is true, the thruster levels are displayed using the values stored. If "draw" is false this step is skipped. The variable "puma" is then checked to see if the propulsion subsystem is on. If "puma" is true, the propulsion system is on and the values stored in the array are transmitted to the propulsion control card using transpuma(). Transpuma() is the analog of transmast(). It insures the propulsion control card is ready and performs all the handshaking necessary to transmit data. It then transmits the data to the propulsion control card. It utilizes the inport() and outport() functions. If "puma" is false (the default), it does not transmit the array values and goes to the next step. The outport function is then called to signal the end of this part of the main loop. The timing test is used to see how long the system is working in the action loop and how long it spends in the option loop described below.

5) The option loop is started by utilizing a screen routine supplied by the compiler. The routine (scrscs()) reads a character in-

there are no hung joint cards and the propulsion card is ready. Upon a response from the stat() function, the action part of the loop begins. The output() function is called to start a timing test. This test is to see how long the central controller is in this part of the main loop. The action part of the loop controls the right arm and the propulsion subsystem by converting the control inputs to commands and transmitting them to the appropriate subsystem.

3) The action part of the loop begins with the control of the right arm. The variable "joyflag" is first tested. It can be true or false (it is initially set to true). If "joyflag" is true, the master arm on the ICS controls the BAT's right arm and the two joysticks control the propulsion subsystem. If "joyflag" is false the two joysticks control the BAT's right arm and the backup controls on the ICS control the propulsion subsystem. If the master arm controls the BAT's arm, the central controller first uses the convert() function to read and convert the analog inputs from the master arm. It then uses the adjust() and align() functions to insure that maximum and minimum limits of the pots are not exceeded and that the pot offsets are included. These values, containing the address, data and commands of each joint/actuator are stored. If "joyflag" is false, the convert() function is used to read the analog inputs from the two 3 degrees-of-freedom joysticks. The joysticks' offsets, computed in calibratebat(), are added and the values are then adjusted. The values for the address, data and commands for each joint/actuator are stored. If the variable "draw" is true, the actuator levels are dis-

- the master arm to control the right arm
- the two 3 degrees-of-freedom joysticks to control the propulsion subsystem
- the propulsion system to off
- the helmet gimbal system to control the video subsystem
- the display of the actuator and thruster levels to off.

It initializes the buffer in the central controller for the control of the BAT's right arm. It does this by initializing three arrays for the address, data and command of each of the joint/actuators on the BAT. It sends these arrays to the buffer/monitor card by using the function `transmast()`. This function is used anytime any information must be passed from the central controller to the buffer/monitor card. It insures that the buffer is not full and that a joint card is not hung up. It performs all handshaking tasks needed to transmit the arrays. `Inport()` and `output()` are used throughout the function. After the buffer/monitor card is initialized, the main loop of `runbat()` is executed.

2) Controlling the main loop is a synchronization function (called `synchronize()`). It does precisely what its name implies and is tested at the start of each loop. If the system is synchronized it continues; if it is not synchronized it exits the loop and goes to `shutdownbat()`. After the synchronization test, the status of the buffer/monitor card and the propulsion control card are checked using the `stat()` function. This function insures that the buffer/monitor card is ready,

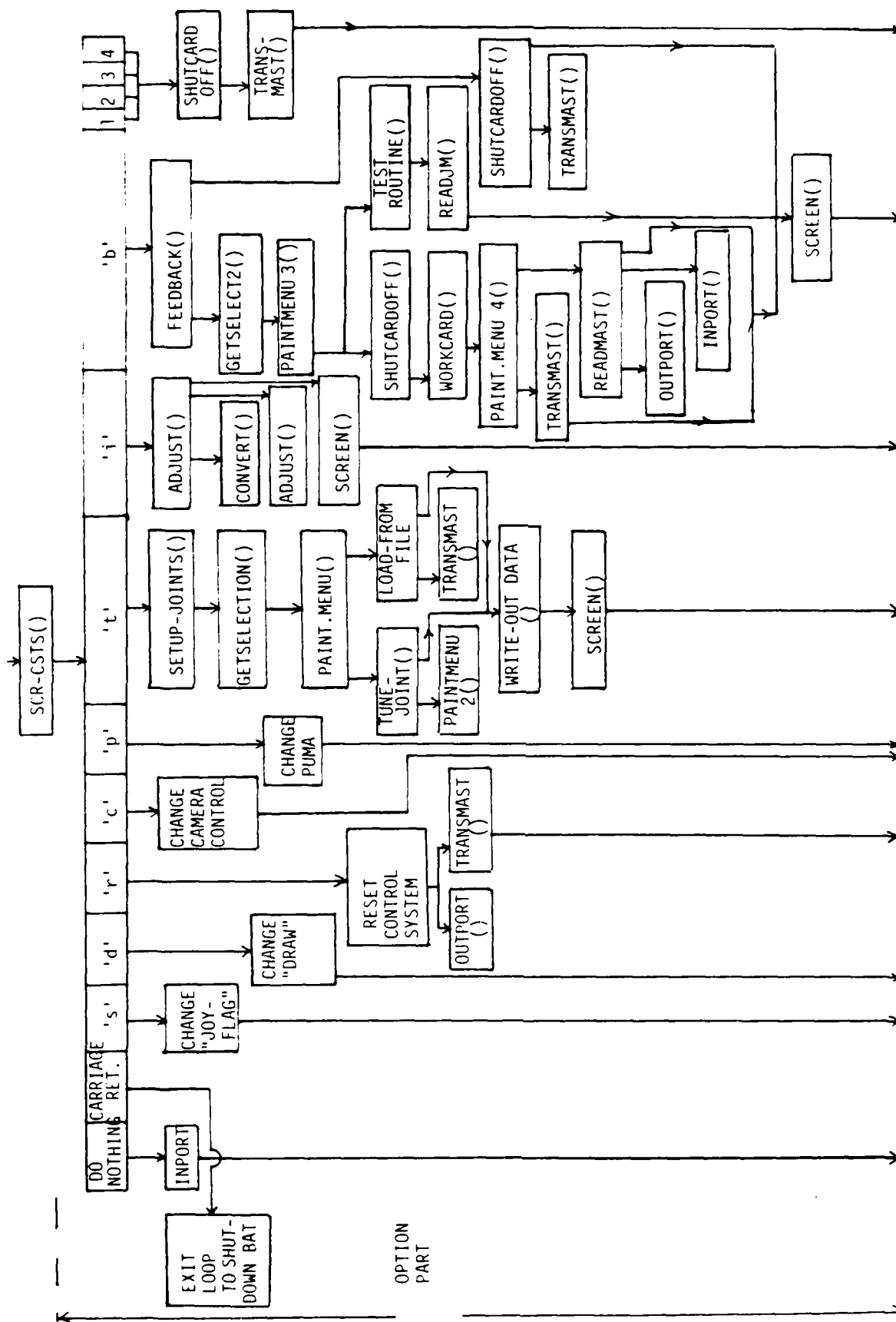


FIGURE A.19b Runbat() Function

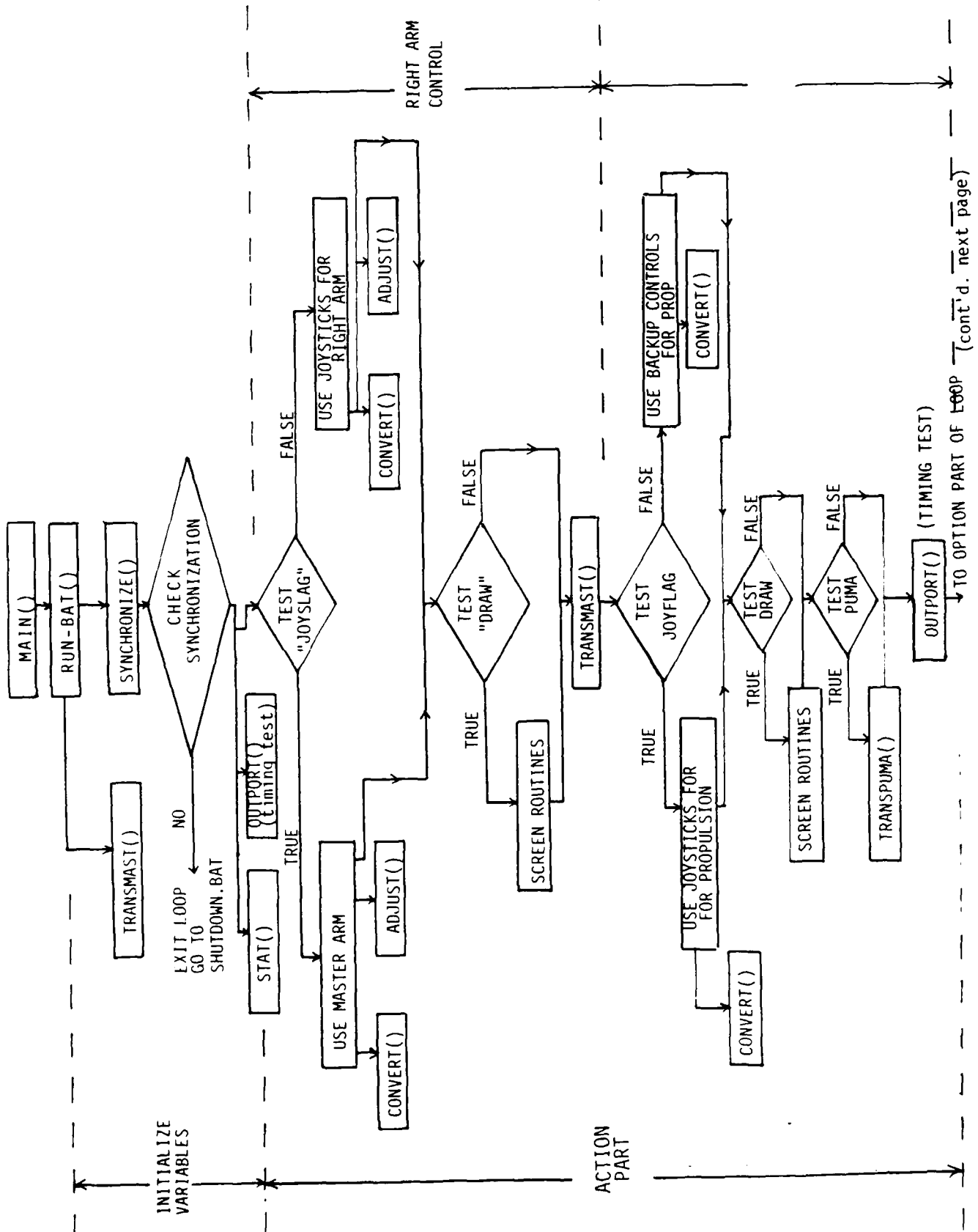


FIGURE A.19a Runbat() Function

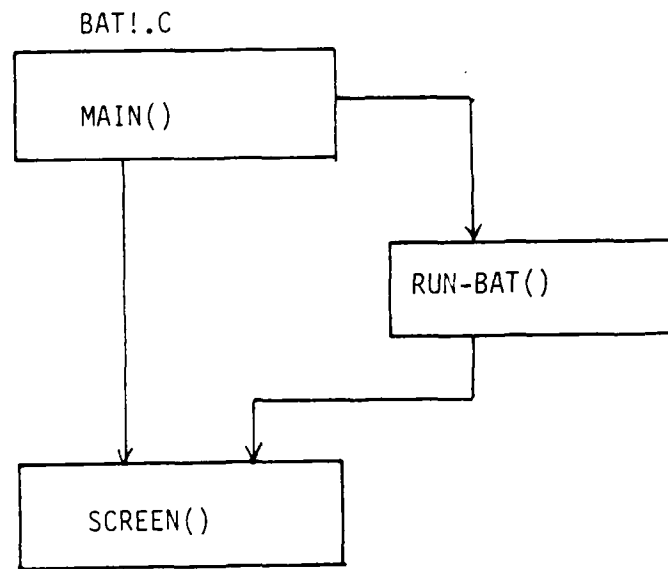


FIGURE A.18 `Screen()` Function

operator to position the joysticks in the center. After waiting for this to be done, it then uses `convert()` for each degree-of-freedom on each joystick. `Convert()` takes an analog input to the central controller's analog-to-digital converter and converts it to an integer value. With the joysticks at the center position and the integer value returned by `convert()`, the central controller computes and stores an offset for each of the joystick's degrees-of-freedom. Control is then returned to `MAIN()`.

C) `screen()` (see Figure A.18)

This function sets up the video background for the displays created by the central controller. It calls screen routines and functions (such as `printf()`) which are implemented in this version of the compiler. The `screen()` function is also called by `runbat()`. `Runbat()` uses some functions which change the display. After these functions are done, they call `screen()` to put up the original display. Control is returned to the calling function after the display is on the screen. Screen routines and input/output functions supplied by the compiler are called throughout the BAT system's software. These routines and the `screen()` function provide the operator with a continuous display of the BAT system operation. The screen routines also provide another method for the operator to interact with the software.

D) `runbat()` (see Figure A.19)

1) After the initialization and calibration functions are completed and the display is set up, the BAT system is ready for operation. The `runbat()` function first initializes the minimum and maximum positions of the potentiometers on the controls and configures the initial controls of the ICS. It sets:

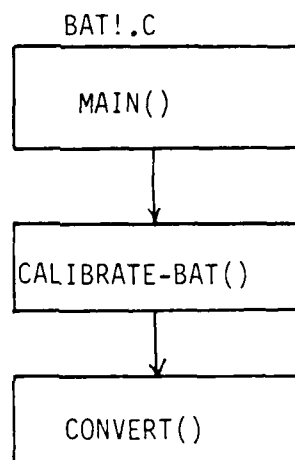


FIGURE A.17 Calibrate Bat() Function

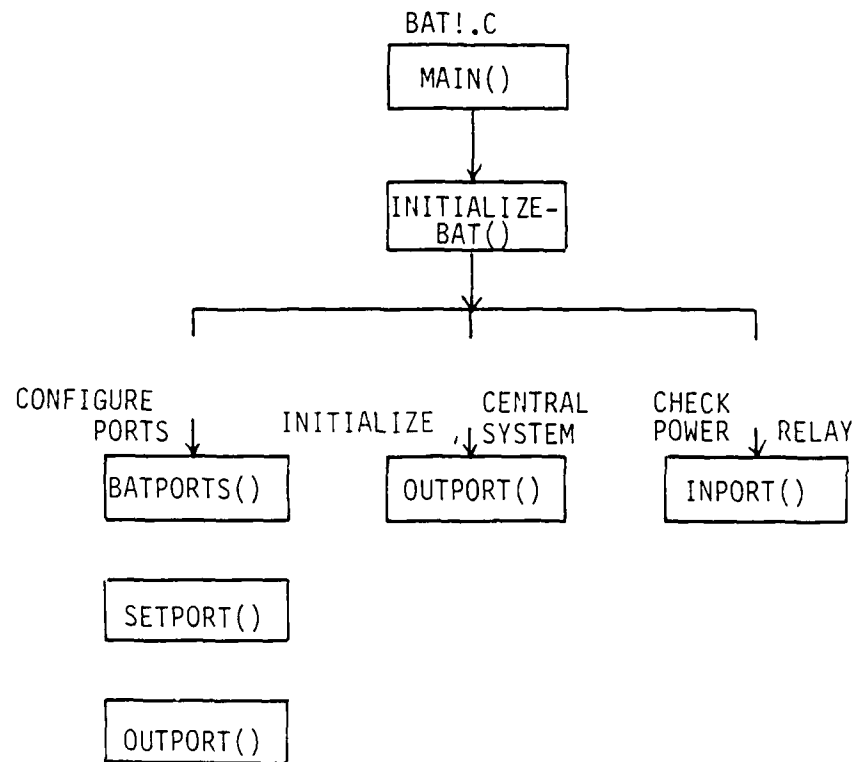


FIGURE A.16 Initialize BAT() Function

-147-

APPENDIX B
PROGRAM FOR SUPERVISORY CONTROL

```

1      /*          SUPERVIS.C
2          */
3      /* This selects the options for supervisory control
4         of the */
5      /* BAT's right arm. It calls move().
6          */
7      /*          Anthony J. Manganiello      Sprin
8         ng 1985      */
9      # define THT1MIN      0.0000001
10     # define THT2MIN      1.3089969
11     # define THT3MIN      0.5235967
12     # define THT4MIN      0.0000001
13     # define THT5MIN      -1.7453292
14     # define THT1MAX      3.1415927
15     # define THT2MAX      2.6179933
16     # define THT3MAX      3.7524578
17     # define THT4MAX      3.1415927
18     # define THT5MAX      1.7453292
19     # define GAMMAX      0.762
20     # define FI      3.1415927
21     # define LI      0.4064
22     # define LZ      0.3556
23     # define AW      0.9310344
24     # include <math.h>
25     superviso()
26     {
27     extern double value[5];
28     double finpos[5] = (2.2689230,.6,-.1,1.5707963,0.0);
29     int intang[5][3];
30     static int angle[7][3] = {
31         {0,0,0,0,0},
32         {0,0,0,0,0},
33         {0,0,0,0,0},
34         {0,0,0,0,0},
35         {0,0,0,0,0},
36         {0,0,0,0,0},
37         {0,0,0,0,0}
38     };
39     int count = 1;
40     int countmax;
41     int i,angle[3] = {0,0,0};
42     while(count)
43     {
44         value[0] = THT1 + THT2 + THT3 + PSI + THT4 + THT5;
45         value[1] = GAM + GAMMAX * angle[0][0] * THT1 * THT2;
46         value[2] = THT1 * THT2;
47         value[3] = THT1 * THT2 * THT3 * THT4 * THT5;
48         value[4] = THT1 * THT2 * THT3 * THT4 * THT5 * GAM;
49         count++;
50     }
51     }

```

```

48 double arcsin();
49 /*****
50 */
51 /*
52 */
53 /*
54 /*****
55 *****/
56 scr_clr();
57 printf("\nSUPERVISORY CONTROL\n\n");
58 printf("OPTIONS:\n");
59 printf("m - move to stored position\n\n");
60 printf("x - store a point for concatenation\n");
61 printf("note- max # of points - 5\n\n");
62 printf("y - clear all stored points\n\n");
63 printf("z - move thru all stored points\n\n");
64 printf("space bar - returns to calling function\n");
65 printf("without selecting an option\n");
66 /*****
67 *****/
68 /-
69 */
70 /*
71 */
72 /*****
73 *****/
74 ready = scr_opts();
75 while(ready == 0)
76     ready = (scr_opts() & 0x20);
77 /*****
78 *****/
79 /-
80 */
81 /*
82 */
83 /*****
84 *****/
85 /-
86 */
87 /*****
88 *****/
89 /-
90 */
91 /*****
92 *****/
93 /-
94 */
95 /*****
96 *****/
97 /-
98 */
99 /*****
100 *****/
101 /-
102 */
103 /*****
104 *****/
105 /-
106 */
107 /*****
108 *****/
109 /-
110 */
111 /*****
112 *****/
113 /-
114 */
115 /*****
116 *****/
117 /-
118 */
119 /*****
120 *****/
121 /-
122 */
123 /*****
124 *****/
125 /-
126 */
127 /*****
128 *****/
129 /-
130 */
131 /*****
132 *****/
133 /-
134 */
135 /*****
136 *****/
137 /-
138 */
139 /*****
140 *****/
141 /-
142 */
143 /*****
144 *****/
145 /-
146 */
147 /*****
148 *****/
149 /-
150 */
151 /*****
152 *****/
153 /-
154 */
155 /*****
156 *****/
157 /-
158 */
159 /*****
160 *****/
161 /-
162 */
163 /*****
164 *****/
165 /-
166 */
167 /*****
168 *****/
169 /-
170 */
171 /*****
172 *****/
173 /-
174 */
175 /*****
176 *****/
177 /-
178 */
179 /*****
180 *****/
181 /-
182 */
183 /*****
184 *****/
185 /-
186 */
187 /*****
188 *****/
189 /-
190 */
191 /*****
192 *****/
193 /-
194 */
195 /*****
196 *****/
197 /-
198 */
199 /*****
200 *****/
201 /-
202 */
203 /*****
204 *****/
205 /-
206 */
207 /*****
208 *****/
209 /-
210 */
211 /*****
212 *****/
213 /-
214 */
215 /*****
216 *****/
217 /-
218 */
219 /*****
220 *****/
221 /-
222 */
223 /*****
224 *****/
225 /-
226 */
227 /*****
228 *****/
229 /-
230 */
231 /*****
232 *****/
233 /-
234 */
235 /*****
236 *****/
237 /-
238 */
239 /*****
240 *****/
241 /-
242 */
243 /*****
244 *****/
245 /-
246 */
247 /*****
248 *****/
249 /-
250 */
251 /*****
252 *****/
253 /-
254 */
255 /*****
256 *****/
257 /-
258 */
259 /*****
260 *****/
261 /-
262 */
263 /*****
264 *****/
265 /-
266 */
267 /*****
268 *****/
269 /-
270 */
271 /*****
272 *****/
273 /-
274 */
275 /*****
276 *****/
277 /-
278 */
279 /*****
280 *****/
281 /-
282 */
283 /*****
284 *****/
285 /-
286 */
287 /*****
288 *****/
289 /-
290 */
291 /*****
292 *****/
293 /-
294 */
295 /*****
296 *****/
297 /-
298 */
299 /*****
300 *****/
301 /-
302 */
303 /*****
304 *****/
305 /-
306 */
307 /*****
308 *****/
309 /-
310 */
311 /*****
312 *****/
313 /-
314 */
315 /*****
316 *****/
317 /-
318 */
319 /*****
320 *****/
321 /-
322 */
323 /*****
324 *****/
325 /-
326 */
327 /*****
328 *****/
329 /-
330 */
331 /*****
332 *****/
333 /-
334 */
335 /*****
336 *****/
337 /-
338 */
339 /*****
340 *****/
341 /-
342 */
343 /*****
344 *****/
345 /-
346 */
347 /*****
348 *****/
349 /-
350 */
351 /*****
352 *****/
353 /-
354 */
355 /*****
356 *****/
357 /-
358 */
359 /*****
360 *****/
361 /-
362 */
363 /*****
364 *****/
365 /-
366 */
367 /*****
368 *****/
369 /-
370 */
371 /*****
372 *****/
373 /-
374 */
375 /*****
376 *****/
377 /-
378 */
379 /*****
380 *****/
381 /-
382 */
383 /*****
384 *****/
385 /-
386 */
387 /*****
388 *****/
389 /-
390 */
391 /*****
392 *****/
393 /-
394 */
395 /*****
396 *****/
397 /-
398 */
399 /*****
400 *****/
401 /-
402 */
403 /*****
404 *****/
405 /-
406 */
407 /*****
408 *****/
409 /-
410 */
411 /*****
412 *****/
413 /-
414 */
415 /*****
416 *****/
417 /-
418 */
419 /*****
420 *****/
421 /-
422 */
423 /*****
424 *****/
425 /-
426 */
427 /*****
428 *****/
429 /-
430 */
431 /*****
432 *****/
433 /-
434 */
435 /*****
436 *****/
437 /-
438 */
439 /*****
440 *****/
441 /-
442 */
443 /*****
444 *****/
445 /-
446 */
447 /*****
448 *****/
449 /-
450 */
451 /*****
452 *****/
453 /-
454 */
455 /*****
456 *****/
457 /-
458 */
459 /*****
460 *****/
461 /-
462 */
463 /*****
464 *****/
465 /-
466 */
467 /*****
468 *****/
469 /-
470 */
471 /*****
472 *****/
473 /-
474 */
475 /*****
476 *****/
477 /-
478 */
479 /*****
480 *****/
481 /-
482 */
483 /*****
484 *****/
485 /-
486 */
487 /*****
488 *****/
489 /-
490 */
491 /*****
492 *****/
493 /-
494 */
495 /*****
496 *****/
497 /-
498 */
499 /*****
500 *****/
501 /-
502 */
503 /*****
504 *****/
505 /-
506 */
507 /*****
508 *****/
509 /-
510 */
511 /*****
512 *****/
513 /-
514 */
515 /*****
516 *****/
517 /-
518 */
519 /*****
520 *****/
521 /-
522 */
523 /*****
524 *****/
525 /-
526 */
527 /*****
528 *****/
529 /-
530 */
531 /*****
532 *****/
533 /-
534 */
535 /*****
536 *****/
537 /-
538 */
539 /*****
540 *****/
541 /-
542 */
543 /*****
544 *****/
545 /-
546 */
547 /*****
548 *****/
549 /-
550 */
551 /*****
552 *****/
553 /-
554 */
555 /*****
556 *****/
557 /-
558 */
559 /*****
560 *****/
561 /-
562 */
563 /*****
564 *****/
565 /-
566 */
567 /*****
568 *****/
569 /-
570 */
571 /*****
572 *****/
573 /-
574 */
575 /*****
576 *****/
577 /-
578 */
579 /*****
580 *****/
581 /-
582 */
583 /*****
584 *****/
585 /-
586 */
587 /*****
588 *****/
589 /-
590 */
591 /*****
592 *****/
593 /-
594 */
595 /*****
596 *****/
597 /-
598 */
599 /*****
600 *****/
601 /-
602 */
603 /*****
604 *****/
605 /-
606 */
607 /*****
608 *****/
609 /-
610 */
611 /*****
612 *****/
613 /-
614 */
615 /*****
616 *****/
617 /-
618 */
619 /*****
620 *****/
621 /-
622 */
623 /*****
624 *****/
625 /-
626 */
627 /*****
628 *****/
629 /-
630 */
631 /*****
632 *****/
633 /-
634 */
635 /*****
636 *****/
637 /-
638 */
639 /*****
640 *****/
641 /-
642 */
643 /*****
644 *****/
645 /-
646 */
647 /*****
648 *****/
649 /-
650 */
651 /*****
652 *****/
653 /-
654 */
655 /*****
656 *****/
657 /-
658 */
659 /*****
660 *****/
661 /-
662 */
663 /*****
664 *****/
665 /-
666 */
667 /*****
668 *****/
669 /-
670 */
671 /*****
672 *****/
673 /-
674 */
675 /*****
676 *****/
677 /-
678 */
679 /*****
680 *****/
681 /-
682 */
683 /*****
684 *****/
685 /-
686 */
687 /*****
688 *****/
689 /-
690 */
691 /*****
692 *****/
693 /-
694 */
695 /*****
696 *****/
697 /-
698 */
699 /*****
700 *****/
701 /-
702 */
703 /*****
704 *****/
705 /-
706 */
707 /*****
708 *****/
709 /-
710 */
711 /*****
712 *****/
713 /-
714 */
715 /*****
716 *****/
717 /-
718 */
719 /*****
720 *****/
721 /-
722 */
723 /*****
724 *****/
725 /-
726 */
727 /*****
728 *****/
729 /-
730 */
731 /*****
732 *****/
733 /-
734 */
735 /*****
736 *****/
737 /-
738 */
739 /*****
740 *****/
741 /-
742 */
743 /*****
744 *****/
745 /-
746 */
747 /*****
748 *****/
749 /-
750 */
751 /*****
752 *****/
753 /-
754 */
755 /*****
756 *****/
757 /-
758 */
759 /*****
760 *****/
761 /-
762 */
763 /*****
764 *****/
765 /-
766 */
767 /*****
768 *****/
769 /-
770 */
771 /*****
772 *****/
773 /-
774 */
775 /*****
776 *****/
777 /-
778 */
779 /*****
780 *****/
781 /-
782 */
783 /*****
784 *****/
785 /-
786 */
787 /*****
788 *****/
789 /-
790 */
791 /*****
792 *****/
793 /-
794 */
795 /*****
796 *****/
797 /-
798 */
799 /*****
800 *****/
801 /-
802 */
803 /*****
804 *****/
805 /-
806 */
807 /*****
808 *****/
809 /-
810 */
811 /*****
812 *****/
813 /-
814 */
815 /*****
816 *****/
817 /-
818 */
819 /*****
820 *****/
821 /-
822 */
823 /*****
824 *****/
825 /-
826 */
827 /*****
828 *****/
829 /-
830 */
831 /*****
832 *****/
833 /-
834 */
835 /*****
836 *****/
837 /-
838 */
839 /*****
840 *****/
841 /-
842 */
843 /*****
844 *****/
845 /-
846 */
847 /*****
848 *****/
849 /-
850 */
851 /*****
852 *****/
853 /-
854 */
855 /*****
856 *****/
857 /-
858 */
859 /*****
860 *****/
861 /-
862 */
863 /*****
864 *****/
865 /-
866 */
867 /*****
868 *****/
869 /-
870 */
871 /*****
872 *****/
873 /-
874 */
875 /*****
876 *****/
877 /-
878 */
879 /*****
880 *****/
881 /-
882 */
883 /*****
884 *****/
885 /-
886 */
887 /*****
888 *****/
889 /-
890 */
891 /*****
892 *****/
893 /-
894 */
895 /*****
896 *****/
897 /-
898 */
899 /*****
900 *****/
901 /-
902 */
903 /*****
904 *****/
905 /-
906 */
907 /*****
908 *****/
909 /-
910 */
911 /*****
912 *****/
913 /-
914 */
915 /*****
916 *****/
917 /-
918 */
919 /*****
920 *****/
921 /-
922 */
923 /*****
924 *****/
925 /-
926 */
927 /*****
928 *****/
929 /-
930 */
931 /*****
932 *****/
933 /-
934 */
935 /*****
936 *****/
937 /-
938 */
939 /*****
940 *****/
941 /-
942 */
943 /*****
944 *****/
945 /-
946 */
947 /*****
948 *****/
949 /-
950 */
951 /*****
952 *****/
953 /-
954 */
955 /*****
956 *****/
957 /-
958 */
959 /*****
960 *****/
961 /-
962 */
963 /*****
964 *****/
965 /-
966 */
967 /*****
968 *****/
969 /-
970 */
971 /*****
972 *****/
973 /-
974 */
975 /*****
976 *****/
977 /-
978 */
979 /*****
980 *****/
981 /-
982 */
983 /*****
984 *****/
985 /-
986 */
987 /*****
988 *****/
989 /-
990 */
991 /*****
992 *****/
993 /-
994 */
995 /*****
996 *****/
997 /-
998 */
999 /*****
1000 *****/

```

```

84         }
85         move_intang,finpos,iedn,icmd);
86         scr_clr();
87         printf("\n TRAJECTORY COMPLETED\n");
88         return;
89     }else{
90         /*****
91         *****/
92         /*
93         */
94         /*****
95         *****/
96         if ready == 1{
97             for i=0,j=1; i=1;i++,j++
98             {
99                 tdat = (convert(list2[i],list3[j]) & 0xFFFF) + 1.2
100                 tdat = adjust(tdat,1);
101                 tdat = align(tdat,1) & 0xFFFF;
102                 anglecount[i,j] = tdat;
103                 count = count + 1;
104                 scr_clr();
105                 printf("POINT STORED\n");
106                 return;
107             }else{
108                 /*****
109                 *****/
110                 /*
111                 */
112                 /*****
113                 *****/
114                 /*
115                 */
116                 /*
117                 */
118                 /*
119                 */
120                 /*
121                 */
122                 /*
123                 */
124                 /*
125                 */
126                 /*
127                 */
128                 /*
129                 */
130                 /*
131                 */
132                 /*
133                 */
134                 /*
135                 */
136                 /*
137                 */
138                 /*
139                 */
140                 /*
141                 */
142                 /*
143                 */
144                 /*
145                 */
146                 /*
147                 */
148                 /*
149                 */
150                 /*
151                 */
152                 /*
153                 */
154                 /*
155                 */
156                 /*
157                 */
158                 /*
159                 */
160                 /*
161                 */
162                 /*
163                 */
164                 /*
165                 */
166                 /*
167                 */
168                 /*
169                 */
170                 /*
171                 */
172                 /*
173                 */
174                 /*
175                 */
176                 /*
177                 */
178                 /*
179                 */
180                 /*
181                 */
182                 /*
183                 */
184                 /*
185                 */
186                 /*
187                 */
188                 /*
189                 */
190                 /*
191                 */
192                 /*
193                 */
194                 /*
195                 */
196                 /*
197                 */
198                 /*
199                 */
200                 /*
201                 */
202                 /*
203                 */
204                 /*
205                 */
206                 /*
207                 */
208                 /*
209                 */
210                 /*
211                 */
212                 /*
213                 */
214                 /*
215                 */
216                 /*
217                 */
218                 /*
219                 */
220                 /*
221                 */
222                 /*
223                 */
224                 /*
225                 */
226                 /*
227                 */
228                 /*
229                 */
230                 /*
231                 */
232                 /*
233                 */
234                 /*
235                 */
236                 /*
237                 */
238                 /*
239                 */
240                 /*
241                 */
242                 /*
243                 */
244                 /*
245                 */
246                 /*
247                 */
248                 /*
249                 */
250                 /*
251                 */
252                 /*
253                 */
254                 /*
255                 */
256                 /*
257                 */
258                 /*
259                 */
260                 /*
261                 */
262                 /*
263                 */
264                 /*
265                 */
266                 /*
267                 */
268                 /*
269                 */
270                 /*
271                 */
272                 /*
273                 */
274                 /*
275                 */
276                 /*
277                 */
278                 /*
279                 */
280                 /*
281                 */
282                 /*
283                 */
284                 /*
285                 */
286                 /*
287                 */
288                 /*
289                 */
290                 /*
291                 */
292                 /*
293                 */
294                 /*
295                 */
296                 /*
297                 */
298                 /*
299                 */
300                 /*
301                 */
302                 /*
303                 */
304                 /*
305                 */
306                 /*
307                 */
308                 /*
309                 */
310                 /*
311                 */
312                 /*
313                 */
314                 /*
315                 */
316                 /*
317                 */
318                 /*
319                 */
320                 /*
321                 */
322                 /*
323                 */
324                 /*
325                 */
326                 /*
327                 */
328                 /*
329                 */
330                 /*
331                 */
332                 /*
333                 */
334                 /*
335                 */
336                 /*
337                 */
338                 /*
339                 */
340                 /*
341                 */
342                 /*
343                 */
344                 /*
345                 */
346                 /*
347                 */
348                 /*
349                 */
350                 /*
351                 */
352                 /*
353                 */
354                 /*
355                 */
356                 /*
357                 */
358                 /*
359                 */
360                 /*
361                 */
362                 /*
363                 */
364                 /*
365                 */
366                 /*
367                 */
368                 /*
369                 */
370                 /*
371                 */
372                 /*
373                 */
374                 /*
375                 */
376                 /*
377                 */
378                 /*
379                 */
380                 /*
381                 */
382                 /*
383                 */
384                 /*
385                 */
386                 /*
387                 */
388                 /*
389                 */
390                 /*
391                 */
392                 /*
393                 */
394                 /*
395                 */
396                 /*
397                 */
398                 /*
399                 */
400                 /*
401                 */
402                 /*
403                 */
404                 /*
405                 */
406                 /*
407                 */
408                 /*
409                 */
410                 /*
411                 */
412                 /*
413                 */
414                 /*
415                 */
416                 /*
417                 */
418                 /*
419                 */
420                 /*
421                 */
422                 /*
423                 */
424                 /*
425                 */
426                 /*
427                 */
428                 /*
429                 */
430                 /*
431                 */
432                 /*
433                 */
434                 /*
435                 */
436                 /*
437                 */
438                 /*
439                 */
440                 /*
441                 */
442                 /*
443                 */
444                 /*
445                 */
446                 /*
447                 */
448                 /*
449                 */
450                 /*
451                 */
452                 /*
453                 */
454                 /*
455                 */
456                 /*
457                 */
458                 /*
459                 */
460                 /*
461                 */
462                 /*
463                 */
464                 /*
465                 */
466                 /*
467                 */
468                 /*
469                 */
470                 /*
471                 */
472                 /*
473                 */
474                 /*
475                 */
476                 /*
477                 */
478                 /*
479                 */
480                 /*
481                 */
482                 /*
483                 */
484                 /*
485                 */
486                 /*
487                 */
488                 /*
489                 */
490                 /*
491                 */
492                 /*
493                 */
494                 /*
495                 */
496                 /*
497                 */
498                 /*
499                 */
500                 /*
501                 */
502                 /*
503                 */
504                 /*
505                 */
506                 /*
507                 */
508                 /*
509                 */
510                 /*
511                 */
512                 /*
513                 */
514                 /*
515                 */
516                 /*
517                 */
518                 /*
519                 */
520                 /*
521                 */
522                 /*
523                 */
524                 /*
525                 */
526                 /*
527                 */
528                 /*
529                 */
530                 /*
531                 */
532                 /*
533                 */
534                 /*
535                 */
536                 /*
537                 */
538                 /*
539                 */
540                 /*
541                 */
542                 /*
543                 */
544                 /*
545                 */
546                 /*
547                 */
548                 /*
549                 */
550                 /*
551                 */
552                 /*
553                 */
554                 /*
555                 */
556                 /*
557                 */
558                 /*
559                 */
560                 /*
561                 */
562                 /*
563                 */
564                 /*
565                 */
566                 /*
567                 */
568                 /*
569                 */
570                 /*
571                 */
572                 /*
573                 */
574                 /*
575                 */
576                 /*
577                 */
578                 /*
579                 */
580                 /*
581                 */
582                 /*
583                 */
584                 /*
585                 */
586                 /*
587                 */
588                 /*
589                 */
590                 /*
591                 */
592                 /*
593                 */
594                 /*
595                 */
596                 /*
597                 */
598                 /*
599                 */
600                 /*
601                 */
602                 /*
603                 */
604                 /*
605                 */
606                 /*
607                 */
608                 /*
609                 */
610                 /*
611                 */
612                 /*
613                 */
614                 /*
615                 */
616                 /*
617                 */
618                 /*
619                 */
620                 /*
621                 */
622                 /*
623                 */
624                 /*
625                 */
626                 /*
627                 */
628                 /*
629                 */
630                 /*
631                 */
632                 /*
633                 */
634                 /*
635                 */
636                 /*
637                 */
638                 /*
639                 */
640                 /*
641                 */
642                 /*
643                 */
644                 /*
645                 */
646                 /*
647                 */
648                 /*
649                 */
650                 /*
651                 */
652                 /*
653                 */
654                 /*
655                 */
656                 /*
657                 */
658                 /*
659                 */
660                 /*
661                 */
662                 /*
663                 */
664                 /*
665                 */
666                 /*
667                 */
668                 /*
669                 */
670                 /*
671                 */
672                 /*
673                 */
674                 /*
675                 */
676                 /*
677                 */
678                 /*
679                 */
680                 /*
681                 */
682                 /*
683                 */
684                 /*
685                 */
686                 /*
687                 */
688                 /*
689                 */
690                 /*
691                 */
692                 /*
693                 */
694                 /*
695                 */
696                 /*
697                 */
698                 /*
699                 */
700                 /*
701                 */
702                 /*
703                 */
704                 /*
705                 */
706                 /*
707                 */
708                 /*
709                 */
710                 /*
711                 */
712                 /*
713                 */
714                 /*
715                 */
716                 /*
717                 */
718                 /*
719                 */
720                 /*
721                 */
722                 /*
723                 */
724                 /*
725                 */
726                 /*
727                 */
728                 /*
729                 */
730                 /*
731                 */
732                 /*
733                 */
734                 /*
735                 */
736                 /*
737                 */
738                 /*
739                 */
740                 /*
741                 */
742                 /*
743                 */
744                 /*
745                 */
746                 /*
747                 */
748                 /*
749                 */
750                 /*
751                 */
752                 /*
753                 */
754                 /*
755                 */
756                 /*
757                 */
758                 /*
759                 */
760                 /*
761                 */
762                 /*
763                 */
764                 /*
765                 */
766                 /*
767                 */
768                 /*
769                 */
770                 /*
771                 */
772                 /*
773                 */
774                 /*
775                 */
776                 /*
777                 */
778                 /*
779                 */
780                 /*
781                 */
782                 /*
783                 */
784                 /*
785                 */
786                 /*
787                 */
788                 /*
789                 */
790                 /*
791                 */
792                 /*
793                 */
794                 /*
795                 */
796                 /*
797                 */
798                 /*
799                 */
800                 /*
801                 */
802                 /*
803                 */
804                 /*
805                 */
806                 /*
807                 */
808                 /*
809                 */
810                 /*
811                 */
812                 /*
813                 */
814                 /*
815                 */
816                 /*
817                 */
818                 /*
819                 */
820                 /*
821                 */
822                 /*
823                 */
824                 /*
825                 */
826                 /*
827                 */
828                 /*
829                 */
830                 /*
831                 */
832                 /*
833                 */
834                 /*
835                 */
836                 /*
837                 */
838                 /*
839                 */
840                 /*
841                 */
842                 /*
843                 */
844                 /*
845                 */
846                 /*
847                 */
848                 /*
849                 */
850                 /*
851                 */
852                 /*
853                 */
854                 /*
855                 */
856                 /*
857                 */
858                 /*
859                 */
860                 /*
861                 */
862                 /*
863                 */
864                 /*
865                 */
866                 /*
867                 */
868                 /*
869                 */
870                 /*
871                 */
872                 /*
873                 */
874                 /*
875                 */
876                 /*
877                 */
878                 /*
879                 */
880                 /*
881                 */
882                 /*
883                 */
884                 /*
885                 */
886                 /*
887                 */
888                 /*
889                 */
890                 /*
891                 */
892                 /*
893                 */
894                 /*
895                 */
896                 /*
897                 */
898                 /*
899                 */
900                 /*
901                 */
902                 /*
903                 */
904                 /*
905                 */
906                 /*
907                 */
908                 /*
909                 */
910                 /*
911                 */
912                 /*
913                 */
914                 /*
915                 */
916                 /*
917                 */
918                 /*
919                 */
920                 /*
921                 */
922                 /*
923                 */
924                 /*
925                 */
926                 /*
927                 */
928                 /*
929                 */
930                 /*
931                 */
932                 /*
933                 */
934                 /*
935                 */
936                 /*
937                 */
938                 /*
939                 */
940                 /*
941                 */
942                 /*
943                 */
944                 /*
945                 */
946                 /*
947                 */
948                 /*
949                 */
950                 /*
951                 */
952                 /*
953                 */
954                 /*
955                 */
956                 /*
957                 */
958                 /*
959                 */
960                 /*
961                 */
962                 /*
963                 */
964                 /*
965                 */
966                 /*
967                 */
968                 /*
969                 */
970                 /*
971                 */
972                 /*
973                 */
974                 /*
975                 */
976                 /*
977                 */
978                 /*
979                 */
980                 /*
981                 */
982                 /*
983                 */
984                 /*
985                 */
986                 /*
987                 */
988                 /*
989                 */
990                 /*
991                 */
992                 /*
993                 */
994                 /*
995                 */
996                 /*
997                 */
998                 /*
999                 */
1000                /*

```



```

161  /******
162  /******
163  TH14F = (-.5*PB14+).5*PB15+.1
164  TH15F = -.1/(2*AW)*PB14F+((1/(2*AW))*PB15F);
165  /******
166  /*
167  /*
168  /*
169  /******
170  if (TH1F > TH1MAX)
171      TH1F = TH1MAX;
172  else if
173      if (TH1F < TH1MIN)
174      TH1F = TH1MIN;
175  }
176  if (TH2F > TH2MAX)
177      TH2F = TH2MAX;
178  else if
179      if (TH2F < TH2MIN)
180      TH2F = TH2MIN;
181  }
182  if (TH3F > TH3MAX)
183      TH3F = TH3MAX;
184  else if
185      if (TH3F < TH3MIN)
186      TH3F = TH3MIN;
187  }
188  if (TH4F > TH4MAX)
189      TH4F = TH4MAX;
190  else if
191      if (TH4F < TH4MIN)
192      TH4F = TH4MIN;
193  }
194  if (TH5F > TH5MAX)
195      TH5F = TH5MAX;
196  else if
197      if (TH5F < TH5MIN)
198      TH5F = TH5MIN;
199  }
200  /******
201  /******
202  /*
203  /*
204  /*
205  /*
206  /*
207  /*
208  /*
209  /*
210  /*
211  /*
212  /*
213  /*
214  /*
215  /*
216  /*
217  /*
218  /*
219  /*
220  /*
221  /*
222  /*
223  /*
224  /*
225  /*
226  /*
227  /*
228  /*
229  /*
230  /*
231  /*
232  /*
233  /*
234  /*
235  /*
236  /*
237  /*
238  /*
239  /*
240  /*
241  /*
242  /*
243  /*
244  /*
245  /*
246  /*
247  /*
248  /*
249  /*
250  /*
251  /*
252  /*
253  /*
254  /*
255  /*
256  /*
257  /*
258  /*
259  /*
260  /*
261  /*
262  /*
263  /*
264  /*
265  /*
266  /*
267  /*
268  /*
269  /*
270  /*
271  /*
272  /*
273  /*
274  /*
275  /*
276  /*
277  /*
278  /*
279  /*
280  /*
281  /*
282  /*
283  /*
284  /*
285  /*
286  /*
287  /*
288  /*
289  /*
290  /*
291  /*
292  /*
293  /*
294  /*
295  /*
296  /*
297  /*
298  /*
299  /*
300  /*
301  /*
302  /*
303  /*
304  /*
305  /*
306  /*
307  /*
308  /*
309  /*
310  /*
311  /*
312  /*
313  /*
314  /*
315  /*
316  /*
317  /*
318  /*
319  /*
320  /*
321  /*
322  /*
323  /*
324  /*
325  /*
326  /*
327  /*
328  /*
329  /*
330  /*
331  /*
332  /*
333  /*
334  /*
335  /*
336  /*
337  /*
338  /*
339  /*
340  /*
341  /*
342  /*
343  /*
344  /*
345  /*
346  /*
347  /*
348  /*
349  /*
350  /*
351  /*
352  /*
353  /*
354  /*
355  /*
356  /*
357  /*
358  /*
359  /*
360  /*
361  /*
362  /*
363  /*
364  /*
365  /*
366  /*
367  /*
368  /*
369  /*
370  /*
371  /*
372  /*
373  /*
374  /*
375  /*
376  /*
377  /*
378  /*
379  /*
380  /*
381  /*
382  /*
383  /*
384  /*
385  /*
386  /*
387  /*
388  /*
389  /*
390  /*
391  /*
392  /*
393  /*
394  /*
395  /*
396  /*
397  /*
398  /*
399  /*
400  /*
401  /*
402  /*
403  /*
404  /*
405  /*
406  /*
407  /*
408  /*
409  /*
410  /*
411  /*
412  /*
413  /*
414  /*
415  /*
416  /*
417  /*
418  /*
419  /*
420  /*
421  /*
422  /*
423  /*
424  /*
425  /*
426  /*
427  /*
428  /*
429  /*
430  /*
431  /*
432  /*
433  /*
434  /*
435  /*
436  /*
437  /*
438  /*
439  /*
440  /*
441  /*
442  /*
443  /*
444  /*
445  /*
446  /*
447  /*
448  /*
449  /*
450  /*
451  /*
452  /*
453  /*
454  /*
455  /*
456  /*
457  /*
458  /*
459  /*
460  /*
461  /*
462  /*
463  /*
464  /*
465  /*
466  /*
467  /*
468  /*
469  /*
470  /*
471  /*
472  /*
473  /*
474  /*
475  /*
476  /*
477  /*
478  /*
479  /*
480  /*
481  /*
482  /*
483  /*
484  /*
485  /*
486  /*
487  /*
488  /*
489  /*
490  /*
491  /*
492  /*
493  /*
494  /*
495  /*
496  /*
497  /*
498  /*
499  /*
500  /*
501  /*
502  /*
503  /*
504  /*
505  /*
506  /*
507  /*
508  /*
509  /*
510  /*
511  /*
512  /*
513  /*
514  /*
515  /*
516  /*
517  /*
518  /*
519  /*
520  /*
521  /*
522  /*
523  /*
524  /*
525  /*
526  /*
527  /*
528  /*
529  /*
530  /*
531  /*
532  /*
533  /*
534  /*
535  /*
536  /*
537  /*
538  /*
539  /*
540  /*
541  /*
542  /*
543  /*
544  /*
545  /*
546  /*
547  /*
548  /*
549  /*
550  /*
551  /*
552  /*
553  /*
554  /*
555  /*
556  /*
557  /*
558  /*
559  /*
560  /*
561  /*
562  /*
563  /*
564  /*
565  /*
566  /*
567  /*
568  /*
569  /*
570  /*
571  /*
572  /*
573  /*
574  /*
575  /*
576  /*
577  /*
578  /*
579  /*
580  /*
581  /*
582  /*
583  /*
584  /*
585  /*
586  /*
587  /*
588  /*
589  /*
590  /*
591  /*
592  /*
593  /*
594  /*
595  /*
596  /*
597  /*
598  /*
599  /*
600  /*
601  /*
602  /*
603  /*
604  /*
605  /*
606  /*
607  /*
608  /*
609  /*
610  /*
611  /*
612  /*
613  /*
614  /*
615  /*
616  /*
617  /*
618  /*
619  /*
620  /*
621  /*
622  /*
623  /*
624  /*
625  /*
626  /*
627  /*
628  /*
629  /*
630  /*
631  /*
632  /*
633  /*
634  /*
635  /*
636  /*
637  /*
638  /*
639  /*
640  /*
641  /*
642  /*
643  /*
644  /*
645  /*
646  /*
647  /*
648  /*
649  /*
650  /*
651  /*
652  /*
653  /*
654  /*
655  /*
656  /*
657  /*
658  /*
659  /*
660  /*
661  /*
662  /*
663  /*
664  /*
665  /*
666  /*
667  /*
668  /*
669  /*
670  /*
671  /*
672  /*
673  /*
674  /*
675  /*
676  /*
677  /*
678  /*
679  /*
680  /*
681  /*
682  /*
683  /*
684  /*
685  /*
686  /*
687  /*
688  /*
689  /*
690  /*
691  /*
692  /*
693  /*
694  /*
695  /*
696  /*
697  /*
698  /*
699  /*
700  /*
701  /*
702  /*
703  /*
704  /*
705  /*
706  /*
707  /*
708  /*
709  /*
710  /*
711  /*
712  /*
713  /*
714  /*
715  /*
716  /*
717  /*
718  /*
719  /*
720  /*
721  /*
722  /*
723  /*
724  /*
725  /*
726  /*
727  /*
728  /*
729  /*
730  /*
731  /*
732  /*
733  /*
734  /*
735  /*
736  /*
737  /*
738  /*
739  /*
740  /*
741  /*
742  /*
743  /*
744  /*
745  /*
746  /*
747  /*
748  /*
749  /*
750  /*
751  /*
752  /*
753  /*
754  /*
755  /*
756  /*
757  /*
758  /*
759  /*
760  /*
761  /*
762  /*
763  /*
764  /*
765  /*
766  /*
767  /*
768  /*
769  /*
770  /*
771  /*
772  /*
773  /*
774  /*
775  /*
776  /*
777  /*
778  /*
779  /*
780  /*
781  /*
782  /*
783  /*
784  /*
785  /*
786  /*
787  /*
788  /*
789  /*
790  /*
791  /*
792  /*
793  /*
794  /*
795  /*
796  /*
797  /*
798  /*
799  /*
800  /*
801  /*
802  /*
803  /*
804  /*
805  /*
806  /*
807  /*
808  /*
809  /*
810  /*
811  /*
812  /*
813  /*
814  /*
815  /*
816  /*
817  /*
818  /*
819  /*
820  /*
821  /*
822  /*
823  /*
824  /*
825  /*
826  /*
827  /*
828  /*
829  /*
830  /*
831  /*
832  /*
833  /*
834  /*
835  /*
836  /*
837  /*
838  /*
839  /*
840  /*
841  /*
842  /*
843  /*
844  /*
845  /*
846  /*
847  /*
848  /*
849  /*
850  /*
851  /*
852  /*

```



```

203 /*
204      */
205 /*****
206 *****/
207 if (THT3f < PI) {
208     THTf = THT1f;
209     GAMMAf = sqrt((L1*L1)+(L2*L2)-(2*L1*L2*cos(THT3f)));
210 }
211 if (GAMMAf > GAMMAX)
212     GAMMAf = GAMMAX;
213 arg1 = (L2*sin(THT3f))/GAMMAf;
214 PHI1 = arcsin(arg1);
215 PSI1 = (PI/2)-(THT2f-PHI1);
216 Zf = GAMMAf*sin(PSI1);
217 Rf = sqrt((GAMMAf*GAMMAf)-(Zf*Zf));
218 }else{
219     if (THT3f > PI) {
220         THTf = THT1f;
221         GAMMAf = sqrt((L1*L1)+(L2*L2)-(2*L1*L2*cos((2+
222 PI)-THT3f)));
223     if (GAMMAf > GAMMAX)
224         GAMMAf = GAMMAX;
225     arg2 = (L2*sin((2*PI)-THT3f))/GAMMAf;
226     PHI2 = arcsin(arg2);
227     PSI2 = (PI/2)-(THT2f+PHI2);
228     Zf = GAMMAf*sin(PSI2);
229     Rf = sqrt((GAMMAf*GAMMAf)-(Zf*Zf));
230 }else{
231     if (THT3f == PI) {
232         THTf = THT1f;
233         GAMMAf = L1+L2;
234         PSI1 = (PI/2)-THT2f;
235         Zf = GAMMAf*(sin(PSI1));
236         Rf = GAMMAf*(cos(PSI1));
237     }else{
238         printf("ERROR IN ENTERING FINAL VALUE OF T
239 HETA 3rd");
240     }
241 }
242 /*****
243 *****/
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```
145  ftopos[4] = THDS+1;
146      move(intang,ftopos,(adr,icmd));
147  }
148  scr_clr();
149  printf("\n TRAJECTORY COMPLETED\n");
150  return;
151  }else{
152      printf("NO SELECTION MADE");    /* Press space bar
153      to exit */
154      return;
155  }
156  }
157  }
158  }
```

```

1      *
2      *
3      * This does the actual driving of the BAT's right &
4      * left wheels.
5      *
6      * Under supervisory control.
7      *
8      *
9      *
10     *
11     *
12     *
13     *
14     *
15     *
16     *
17     *
18     *
19     *
20     *
21     *
22     *
23     *
24     *
25     *
26     *
27     *
28     *
29     *
30     *
31     *
32     *
33     *
34     *
35     *
36     *
37     *
38     *
39     *
40     *
41     *
42     *
43     *
44     *
45     *
46     *
47     *
48     *
49     *
50     *
51     *
52     *
53     *
54     *
55     *
56     *
57     *
58     *
59     *
60     *
61     *
62     *
63     *
64     *
65     *
66     *
67     *
68     *
69     *
70     *
71     *
72     *
73     *
74     *
75     *
76     *
77     *
78     *
79     *
80     *
81     *
82     *
83     *
84     *
85     *
86     *
87     *
88     *
89     *
90     *
91     *
92     *
93     *
94     *
95     *
96     *
97     *
98     *
99     *
100    *
101    *
102    *
103    *
104    *
105    *
106    *
107    *
108    *
109    *
110    *
111    *
112    *
113    *
114    *
115    *
116    *
117    *
118    *
119    *
120    *
121    *
122    *
123    *
124    *
125    *
126    *
127    *
128    *
129    *
130    *
131    *
132    *
133    *
134    *
135    *
136    *
137    *
138    *
139    *
140    *
141    *
142    *
143    *
144    *
145    *
146    *
147    *
148    *
149    *
150    *
151    *
152    *
153    *
154    *
155    *
156    *
157    *
158    *
159    *
160    *
161    *
162    *
163    *
164    *
165    *
166    *
167    *
168    *
169    *
170    *
171    *
172    *
173    *
174    *
175    *
176    *
177    *
178    *
179    *
180    *
181    *
182    *
183    *
184    *
185    *
186    *
187    *
188    *
189    *
190    *
191    *
192    *
193    *
194    *
195    *
196    *
197    *
198    *
199    *
200    *
201    *
202    *
203    *
204    *
205    *
206    *
207    *
208    *
209    *
210    *
211    *
212    *
213    *
214    *
215    *
216    *
217    *
218    *
219    *
220    *
221    *
222    *
223    *
224    *
225    *
226    *
227    *
228    *
229    *
230    *
231    *
232    *
233    *
234    *
235    *
236    *
237    *
238    *
239    *
240    *
241    *
242    *
243    *
244    *
245    *
246    *
247    *
248    *
249    *
250    *
251    *
252    *
253    *
254    *
255    *
256    *
257    *
258    *
259    *
260    *
261    *
262    *
263    *
264    *
265    *
266    *
267    *
268    *
269    *
270    *
271    *
272    *
273    *
274    *
275    *
276    *
277    *
278    *
279    *
280    *
281    *
282    *
283    *
284    *
285    *
286    *
287    *
288    *
289    *
290    *
291    *
292    *
293    *
294    *
295    *
296    *
297    *
298    *
299    *
300    *
301    *
302    *
303    *
304    *
305    *
306    *
307    *
308    *
309    *
310    *
311    *
312    *
313    *
314    *
315    *
316    *
317    *
318    *
319    *
320    *
321    *
322    *
323    *
324    *
325    *
326    *
327    *
328    *
329    *
330    *
331    *
332    *
333    *
334    *
335    *
336    *
337    *
338    *
339    *
340    *
341    *
342    *
343    *
344    *
345    *
346    *
347    *
348    *
349    *
350    *
351    *
352    *
353    *
354    *
355    *
356    *
357    *
358    *
359    *
360    *
361    *
362    *
363    *
364    *
365    *
366    *
367    *
368    *
369    *
370    *
371    *
372    *
373    *
374    *
375    *
376    *
377    *
378    *
379    *
380    *
381    *
382    *
383    *
384    *
385    *
386    *
387    *
388    *
389    *
390    *
391    *
392    *
393    *
394    *
395    *
396    *
397    *
398    *
399    *
400    *
401    *
402    *
403    *
404    *
405    *
406    *
407    *
408    *
409    *
410    *
411    *
412    *
413    *
414    *
415    *
416    *
417    *
418    *
419    *
420    *
421    *
422    *
423    *
424    *
425    *
426    *
427    *
428    *
429    *
430    *
431    *
432    *
433    *
434    *
435    *
436    *
437    *
438    *
439    *
440    *
441    *
442    *
443    *
444    *
445    *
446    *
447    *
448    *
449    *
450    *
451    *
452    *
453    *
454    *
455    *
456    *
457    *
458    *
459    *
460    *
461    *
462    *
463    *
464    *
465    *
466    *
467    *
468    *
469    *
470    *
471    *
472    *
473    *
474    *
475    *
476    *
477    *
478    *
479    *
480    *
481    *
482    *
483    *
484    *
485    *
486    *
487    *
488    *
489    *
490    *
491    *
492    *
493    *
494    *
495    *
496    *
497    *
498    *
499    *
500    *
501    *
502    *
503    *
504    *
505    *
506    *
507    *
508    *
509    *
510    *
511    *
512    *
513    *
514    *
515    *
516    *
517    *
518    *
519    *
520    *
521    *
522    *
523    *
524    *
525    *
526    *
527    *
528    *
529    *
530    *
531    *
532    *
533    *
534    *
535    *
536    *
537    *
538    *
539    *
540    *
541    *
542    *
543    *
544    *
545    *
546    *
547    *
548    *
549    *
550    *
551    *
552    *
553    *
554    *
555    *
556    *
557    *
558    *
559    *
560    *
561    *
562    *
563    *
564    *
565    *
566    *
567    *
568    *
569    *
570    *
571    *
572    *
573    *
574    *
575    *
576    *
577    *
578    *
579    *
580    *
581    *
582    *
583    *
584    *
585    *
586    *
587    *
588    *
589    *
590    *
591    *
592    *
593    *
594    *
595    *
596    *
597    *
598    *
599    *
600    *
601    *
602    *
603    *
604    *
605    *
606    *
607    *
608    *
609    *
610    *
611    *
612    *
613    *
614    *
615    *
616    *
617    *
618    *
619    *
620    *
621    *
622    *
623    *
624    *
625    *
626    *
627    *
628    *
629    *
630    *
631    *
632    *
633    *
634    *
635    *
636    *
637    *
638    *
639    *
640    *
641    *
642    *
643    *
644    *
645    *
646    *
647    *
648    *
649    *
650    *
651    *
652    *
653    *
654    *
655    *
656    *
657    *
658    *
659    *
660    *
661    *
662    *
663    *
664    *
665    *
666    *
667    *
668    *
669    *
670    *
671    *
672    *
673    *
674    *
675    *
676    *
677    *
678    *
679    *
680    *
681    *
682    *
683    *
684    *
685    *
686    *
687    *
688    *
689    *
690    *
691    *
692    *
693    *
694    *
695    *
696    *
697    *
698    *
699    *
700    *
701    *
702    *
703    *
704    *
705    *
706    *
707    *
708    *
709    *
710    *
711    *
712    *
713    *
714    *
715    *
716    *
717    *
718    *
719    *
720    *
721    *
722    *
723    *
724    *
725    *
726    *
727    *
728    *
729    *
730    *
731    *
732    *
733    *
734    *
735    *
736    *
737    *
738    *
739    *
740    *
741    *
742    *
743    *
744    *
745    *
746    *
747    *
748    *
749    *
750    *
751    *
752    *
753    *
754    *
755    *
756    *
757    *
758    *
759    *
760    *
761    *
762    *
763    *
764    *
765    *
766    *
767    *
768    *
769    *
770    *
771    *
772    *
773    *
774    *
775    *
776    *
777    *
778    *
779    *
780    *
781    *
782    *
783    *
784    *
785    *
786    *
787    *
788    *
789    *
790    *
791    *
792    *
793    *
794    *
795    *
796    *
797    *
798    *
799    *
800    *
801    *
802    *
803    *
804    *
805    *
806    *
807    *
808    *
809    *
810    *
811    *
812    *
813    *
814    *
815    *
816    *
817    *
818    *
819    *
820    *
821    *
822    *
823    *
824    *
825    *
826    *
827    *
828    *
829    *
830    *
831    *
832    *
833    *
834    *
835    *
836    *
837    *
838    *
839    *
840    *
841    *
842    *
843    *
844    *
845    *
846    *
847    *
848    *
849    *
850    *
851    *
852    *
853    *
854    *
855    *
856    *
857    *
858    *
859    *
860    *
861    *
862    *
863    *
864    *
865    *
866    *
867    *
868    *
869    *
870    *
871    *
872    *
873    *
874    *
875    *
876    *
877    *
878    *
879    *
880    *
881    *
882    *
883    *
884    *
885    *
886    *
887    *
888    *
889    *
890    *
891    *
892    *
893    *
894    *
895    *
896    *
897    *
898    *
899    *
900    *
901    *
902    *
903    *
904    *
905    *
906    *
907    *
908    *
909    *
910    *
911    *
912    *
913    *
914    *
915    *
916    *
917    *
918    *
919    *
920    *
921    *
922    *
923    *
924    *
925    *
926    *
927    *
928    *
929    *
930    *
931    *
932    *
933    *
934    *
935    *
936    *
937    *
938    *
939    *
940    *
941    *
942    *
943    *
944    *
945    *
946    *
947    *
948    *
949    *
950    *
951    *
952    *
953    *
954    *
955    *
956    *
957    *
958    *
959    *
960    *
961    *
962    *
963    *
964    *
965    *
966    *
967    *
968    *
969    *
970    *
971    *
972    *
973    *
974    *
975    *
976    *
977    *
978    *
979    *
980    *
981    *
982    *
983    *
984    *
985    *
986    *
987    *
988    *
989    *
990    *
991    *
992    *
993    *
994    *
995    *
996    *
997    *
998    *
999    *
1000    *

```

```

47      *
48      /-----/
49      *****/
50      cis(vintang);
51      TH11 = value[0];
52      TH21 = value[1];
53      TH31 = value[2];
54      PSI41 = value[3];
55      PSI51 = value[4];
56      /-----/
57      *****/
58      *
59      *
60      *
61      *
62      *
63      *
64      *
65      *
66      *
67      *
68      *
69      *
70      *
71      *
72      *
73      *
74      *
75      *
76      *
77      *
78      *
79      *
80      *
81      *
82      *
83      *
84      *
85      *
86      *
87      *
88      *
89      *
90      *
91      *
92      *
93      *
94      *
95      *
96      *
97      *
98      *
99      *
100     *
101     *
102     *
103     *
104     *
105     *
106     *
107     *
108     *
109     *
110     *
111     *
112     *
113     *
114     *
115     *
116     *
117     *
118     *
119     *
120     *
121     *
122     *
123     *
124     *
125     *
126     *
127     *
128     *
129     *
130     *
131     *
132     *
133     *
134     *
135     *
136     *
137     *
138     *
139     *
140     *
141     *
142     *
143     *
144     *
145     *
146     *
147     *
148     *
149     *
150     *
151     *
152     *
153     *
154     *
155     *
156     *
157     *
158     *
159     *
160     *
161     *
162     *
163     *
164     *
165     *
166     *
167     *
168     *
169     *
170     *
171     *
172     *
173     *
174     *
175     *
176     *
177     *
178     *
179     *
180     *
181     *
182     *
183     *
184     *
185     *
186     *
187     *
188     *
189     *
190     *
191     *
192     *
193     *
194     *
195     *
196     *
197     *
198     *
199     *
200     *
201     *
202     *
203     *
204     *
205     *
206     *
207     *
208     *
209     *
210     *
211     *
212     *
213     *
214     *
215     *
216     *
217     *
218     *
219     *
220     *
221     *
222     *
223     *
224     *
225     *
226     *
227     *
228     *
229     *
230     *
231     *
232     *
233     *
234     *
235     *
236     *
237     *
238     *
239     *
240     *
241     *
242     *
243     *
244     *
245     *
246     *
247     *
248     *
249     *
250     *
251     *
252     *
253     *
254     *
255     *
256     *
257     *
258     *
259     *
260     *
261     *
262     *
263     *
264     *
265     *
266     *
267     *
268     *
269     *
270     *
271     *
272     *
273     *
274     *
275     *
276     *
277     *
278     *
279     *
280     *
281     *
282     *
283     *
284     *
285     *
286     *
287     *
288     *
289     *
290     *
291     *
292     *
293     *
294     *
295     *
296     *
297     *
298     *
299     *
300     *
301     *
302     *
303     *
304     *
305     *
306     *
307     *
308     *
309     *
310     *
311     *
312     *
313     *
314     *
315     *
316     *
317     *
318     *
319     *
320     *
321     *
322     *
323     *
324     *
325     *
326     *
327     *
328     *
329     *
330     *
331     *
332     *
333     *
334     *
335     *
336     *
337     *
338     *
339     *
340     *
341     *
342     *
343     *
344     *
345     *
346     *
347     *
348     *
349     *
350     *
351     *
352     *
353     *
354     *
355     *
356     *
357     *
358     *
359     *
360     *
361     *
362     *
363     *
364     *
365     *
366     *
367     *
368     *
369     *
370     *
371     *
372     *
373     *
374     *
375     *
376     *
377     *
378     *
379     *
380     *
381     *
382     *
383     *
384     *
385     *
386     *
387     *
388     *
389     *
390     *
391     *
392     *
393     *
394     *
395     *
396     *
397     *
398     *
399     *
400     *
401     *
402     *
403     *
404     *
405     *
406     *
407     *
408     *
409     *
410     *
411     *
412     *
413     *
414     *
415     *
416     *
417     *
418     *
419     *
420     *
421     *
422     *
423     *
424     *
425     *
426     *
427     *
428     *
429     *
430     *
431     *
432     *
433     *
434     *
435     *
436     *
437     *
438     *
439     *
440     *
441     *
442     *
443     *
444     *
445     *
446     *
447     *
448     *
449     *
450     *
451     *
452     *
453     *
454     *
455     *
456     *
457     *
458     *
459     *
460     *
461     *
462     *
463     *
464     *
465     *
466     *
467     *
468     *
469     *
470     *
471     *
472     *
473     *
474     *
475     *
476     *
477     *
478     *
479     *
480     *
481     *
482     *
483     *
484     *
485     *
486     *
487     *
488     *
489     *
490     *
491     *
492     *
493     *
494     *
495     *
496     *
497     *
498     *
499     *
500     *
501     *
502     *
503     *
504     *
505     *
506     *
507     *
508     *
509     *
510     *
511     *
512     *
513     *
514     *
515     *
516     *
517     *
518     *
519     *
520     *
521     *
522     *
523     *
524     *
525     *
526     *
527     *
528     *
529     *
530     *
531     *
532     *
533     *
534     *
535     *
536     *
537     *
538     *
539     *
540     *
541     *
542     *
543     *
544     *
545     *
546     *
547     *
548     *
549     *
550     *
551     *
552     *
553     *
554     *
555     *
556     *
557     *
558     *
559     *
560     *
561     *
562     *
563     *
564     *
565     *
566     *
567     *
568     *
569     *
570     *
571     *
572     *
573     *
574     *
575     *
576     *
577     *
578     *
579     *
580     *
581     *
582     *
583     *
584     *
585     *
586     *
587     *
588     *
589     *
590     *
591     *
592     *
593     *
594     *
595     *
596     *
597     *
598     *
599     *
600     *
601     *
602     *
603     *
604     *
605     *
606     *
607     *
608     *
609     *
610     *
611     *
612     *
613     *
614     *
615     *
616     *
617     *
618     *
619     *
620     *
621     *
622     *
623     *
624     *
625     *
626     *
627     *
628     *
629     *
630     *
631     *
632     *
633     *
634     *
635     *
636     *
637     *
638     *
639     *
640     *
641     *
642     *
643     *
644     *
645     *
646     *
647     *
648     *
649     *
650     *
651     *
652     *
653     *
654     *
655     *
656     *
657     *
658     *
659     *
660     *
661     *
662     *
663     *
664     *
665     *
666     *
667     *
668     *
669     *
670     *
671     *
672     *
673     *
674     *
675     *
676     *
677     *
678     *
679     *
680     *
681     *
682     *
683     *
684     *
685     *
686     *
687     *
688     *
689     *
690     *
691     *
692     *
693     *
694     *
695     *
696     *
697     *
698     *
699     *
700     *
701     *
702     *
703     *
704     *
705     *
706     *
707     *
708     *
709     *
710     *
711     *
712     *
713     *
714     *
715     *
716     *
717     *
718     *
719     *
720     *
721     *
722     *
723     *
724     *
725     *
726     *
727     *
728     *
729     *
730
```

```

85      if (THT41 > THT4MAX)
86          THT41 = THT4MAX;
87      elseif
88          if (THT41 < THT4MIN)
89              THT41 = THT4MIN;
90      }
91      if (THT51 > THT5MAX)
92          THT51 = THT5MAX;
93      elseif
94          if (THT51 < THT5MIN)
95              THT51 = THT5MIN;
96      }
97      /*****
98      *****/
99      /*
100      */
101      /*****
102      *****/
103      if (THT31 < PI) {
104          a = 0;
105          THT1 = THT11;
106          GAMMA1 = sqrt((L1*L1)+(L2*L2)-(2*L1*L2*cos(THT31)));
107          if (GAMMA1 > GAMMAX)
108              GAMMA1 = GAMMAX;
109          arg1 = (L2*sin(THT31))/GAMMA1;
110          PHI1 = arcsin(arg1);
111          PSI1 = (PI/2)-(THT21-PHI1);
112          Zi = GAMMA1*sin(PSI1);
113          Ri = sqrt((GAMMA1+GAMMA1)-(Zi*Zi));
114      } else {
115          if (THT31 > PI) {
116              a = 1;
117              THT1 = THT11;
118              GAMMA1 = sqrt((L1+L1)+(L2+L2)-(2*L1*L2*cos((2*
119      PI)-THT31)));
120          } else {
121              GAMMA1 = GAMMA1;
122              GAMMA1 = GAMMA1;
123              arg1 = (L2*sin((2*PI)-THT31))/GAMMA1;
124              PHI1 = arcsin(arg1);
125              PSI1 = (PI/2)-(THT21+PHI1);
126              Zi = GAMMA1*sin(PSI1);
127              Ri = sqrt((GAMMA1+GAMMA1)-(Zi*Zi));
128          }
129          if (PHI1 > PI)
130              PHI1 = PHI1-PI;
131          THT1 = THT11;
132          GAMMA1 = L1+L2;

```

```

130          PSI1 = (PI(2)-THI21);
131          Zi = GAMMA1*(sin(PSI1));
132          Ri = GAMMA1*(cos(PSI1));
133          Jalse1
134          print("ERROR IN ENTERING INITIAL VALUE OF
      THETA TH1");
135      ) )
136      /*****
      *****/
137      /*
      */
138      /* INPUT THE FINAL POSITION IN CYLINDRICAL COORDIN
      AIES */
139      /*
      */
140      /*****
      *****/
141      THF = finpos(0);
142      Rf = finpos(1);
143      Zf = finpos(2);
144      THI4 = finpos(3);
145      THF5 = finpos(4);
146      /*****
      *****/
147      /*
      */
148      /* COMPUTE CONSTANTS
      */
149      /*
      */
150      /*****
      *****/
151      U = Rf*cos(THF);
152      V = Rf*sin(THF);
153      W = Zf;
154      U4 = Rf*cos(THI4);
155      V4 = Rf*sin(THI4);
156      W4 = Zf;
157      U5 = Rf*cos(THF5);
158      V5 = Rf*sin(THF5);
159      W5 = Zf;
160      U6 = Rf*cos(THI6);
161      V6 = Rf*sin(THI6);
162      W6 = Zf;
163      U7 = Rf*cos(THI7);
164      V7 = Rf*sin(THI7);
165      W7 = Zf;
166      U8 = Rf*cos(THI8);
167      V8 = Rf*sin(THI8);
168      W8 = Zf;
169      U9 = Rf*cos(THI9);
170      V9 = Rf*sin(THI9);
171      W9 = Zf;
172      U10 = Rf*cos(THI10);
173      V10 = Rf*sin(THI10);
174      W10 = Zf;
175      U11 = Rf*cos(THI11);
176      V11 = Rf*sin(THI11);
177      W11 = Zf;
178      U12 = Rf*cos(THI12);
179      V12 = Rf*sin(THI12);
180      W12 = Zf;
181      U13 = Rf*cos(THI13);
182      V13 = Rf*sin(THI13);
183      W13 = Zf;
184      U14 = Rf*cos(THI14);
185      V14 = Rf*sin(THI14);
186      W14 = Zf;
187      U15 = Rf*cos(THI15);
188      V15 = Rf*sin(THI15);
189      W15 = Zf;
190      U16 = Rf*cos(THI16);
191      V16 = Rf*sin(THI16);
192      W16 = Zf;
193      U17 = Rf*cos(THI17);
194      V17 = Rf*sin(THI17);
195      W17 = Zf;
196      U18 = Rf*cos(THI18);
197      V18 = Rf*sin(THI18);
198      W18 = Zf;
199      U19 = Rf*cos(THI19);
200      V19 = Rf*sin(THI19);
201      W19 = Zf;
202      U20 = Rf*cos(THI20);
203      V20 = Rf*sin(THI20);
204      W20 = Zf;
205      U21 = Rf*cos(THI21);
206      V21 = Rf*sin(THI21);
207      W21 = Zf;
208      U22 = Rf*cos(THI22);
209      V22 = Rf*sin(THI22);
210      W22 = Zf;
211      U23 = Rf*cos(THI23);
212      V23 = Rf*sin(THI23);
213      W23 = Zf;
214      U24 = Rf*cos(THI24);
215      V24 = Rf*sin(THI24);
216      W24 = Zf;
217      U25 = Rf*cos(THI25);
218      V25 = Rf*sin(THI25);
219      W25 = Zf;
220      U26 = Rf*cos(THI26);
221      V26 = Rf*sin(THI26);
222      W26 = Zf;
223      U27 = Rf*cos(THI27);
224      V27 = Rf*sin(THI27);
225      W27 = Zf;
226      U28 = Rf*cos(THI28);
227      V28 = Rf*sin(THI28);
228      W28 = Zf;
229      U29 = Rf*cos(THI29);
230      V29 = Rf*sin(THI29);
231      W29 = Zf;
232      U30 = Rf*cos(THI30);
233      V30 = Rf*sin(THI30);
234      W30 = Zf;
235      U31 = Rf*cos(THI31);
236      V31 = Rf*sin(THI31);
237      W31 = Zf;
238      U32 = Rf*cos(THI32);
239      V32 = Rf*sin(THI32);
240      W32 = Zf;
241      U33 = Rf*cos(THI33);
242      V33 = Rf*sin(THI33);
243      W33 = Zf;
244      U34 = Rf*cos(THI34);
245      V34 = Rf*sin(THI34);
246      W34 = Zf;
247      U35 = Rf*cos(THI35);
248      V35 = Rf*sin(THI35);
249      W35 = Zf;
250      U36 = Rf*cos(THI36);
251      V36 = Rf*sin(THI36);
252      W36 = Zf;
253      U37 = Rf*cos(THI37);
254      V37 = Rf*sin(THI37);
255      W37 = Zf;
256      U38 = Rf*cos(THI38);
257      V38 = Rf*sin(THI38);
258      W38 = Zf;
259      U39 = Rf*cos(THI39);
260      V39 = Rf*sin(THI39);
261      W39 = Zf;
262      U40 = Rf*cos(THI40);
263      V40 = Rf*sin(THI40);
264      W40 = Zf;
265      U41 = Rf*cos(THI41);
266      V41 = Rf*sin(THI41);
267      W41 = Zf;
268      U42 = Rf*cos(THI42);
269      V42 = Rf*sin(THI42);
270      W42 = Zf;
271      U43 = Rf*cos(THI43);
272      V43 = Rf*sin(THI43);
273      W43 = Zf;
274      U44 = Rf*cos(THI44);
275      V44 = Rf*sin(THI44);
276      W44 = Zf;
277      U45 = Rf*cos(THI45);
278      V45 = Rf*sin(THI45);
279      W45 = Zf;
280      U46 = Rf*cos(THI46);
281      V46 = Rf*sin(THI46);
282      W46 = Zf;
283      U47 = Rf*cos(THI47);
284      V47 = Rf*sin(THI47);
285      W47 = Zf;
286      U48 = Rf*cos(THI48);
287      V48 = Rf*sin(THI48);
288      W48 = Zf;
289      U49 = Rf*cos(THI49);
290      V49 = Rf*sin(THI49);
291      W49 = Zf;
292      U50 = Rf*cos(THI50);
293      V50 = Rf*sin(THI50);
294      W50 = Zf;
295      U51 = Rf*cos(THI51);
296      V51 = Rf*sin(THI51);
297      W51 = Zf;
298      U52 = Rf*cos(THI52);
299      V52 = Rf*sin(THI52);
300      W52 = Zf;
301      U53 = Rf*cos(THI53);
302      V53 = Rf*sin(THI53);
303      W53 = Zf;
304      U54 = Rf*cos(THI54);
305      V54 = Rf*sin(THI54);
306      W54 = Zf;
307      U55 = Rf*cos(THI55);
308      V55 = Rf*sin(THI55);
309      W55 = Zf;
310      U56 = Rf*cos(THI56);
311      V56 = Rf*sin(THI56);
312      W56 = Zf;
313      U57 = Rf*cos(THI57);
314      V57 = Rf*sin(THI57);
315      W57 = Zf;
316      U58 = Rf*cos(THI58);
317      V58 = Rf*sin(THI58);
318      W58 = Zf;
319      U59 = Rf*cos(THI59);
320      V59 = Rf*sin(THI59);
321      W59 = Zf;
322      U60 = Rf*cos(THI60);
323      V60 = Rf*sin(THI60);
324      W60 = Zf;
325      U61 = Rf*cos(THI61);
326      V61 = Rf*sin(THI61);
327      W61 = Zf;
328      U62 = Rf*cos(THI62);
329      V62 = Rf*sin(THI62);
330      W62 = Zf;
331      U63 = Rf*cos(THI63);
332      V63 = Rf*sin(THI63);
333      W63 = Zf;
334      U64 = Rf*cos(THI64);
335      V64 = Rf*sin(THI64);
336      W64 = Zf;
337      U65 = Rf*cos(THI65);
338      V65 = Rf*sin(THI65);
339      W65 = Zf;
340      U66 = Rf*cos(THI66);
341      V66 = Rf*sin(THI66);
342      W66 = Zf;
343      U67 = Rf*cos(THI67);
344      V67 = Rf*sin(THI67);
345      W67 = Zf;
346      U68 = Rf*cos(THI68);
347      V68 = Rf*sin(THI68);
348      W68 = Zf;
349      U69 = Rf*cos(THI69);
350      V69 = Rf*sin(THI69);
351      W69 = Zf;
352      U70 = Rf*cos(THI70);
353      V70 = Rf*sin(THI70);
354      W70 = Zf;
355      U71 = Rf*cos(THI71);
356      V71 = Rf*sin(THI71);
357      W71 = Zf;
358      U72 = Rf*cos(THI72);
359      V72 = Rf*sin(THI72);
360      W72 = Zf;
361      U73 = Rf*cos(THI73);
362      V73 = Rf*sin(THI73);
363      W73 = Zf;
364      U74 = Rf*cos(THI74);
365      V74 = Rf*sin(THI74);
366      W74 = Zf;
367      U75 = Rf*cos(THI75);
368      V75 = Rf*sin(THI75);
369      W75 = Zf;
370      U76 = Rf*cos(THI76);
371      V76 = Rf*sin(THI76);
372      W76 = Zf;
373      U77 = Rf*cos(THI77);
374      V77 = Rf*sin(THI77);
375      W77 = Zf;
376      U78 = Rf*cos(THI78);
377      V78 = Rf*sin(THI78);
378      W78 = Zf;
379      U79 = Rf*cos(THI79);
380      V79 = Rf*sin(THI79);
381      W79 = Zf;
382      U80 = Rf*cos(THI80);
383      V80 = Rf*sin(THI80);
384      W80 = Zf;
385      U81 = Rf*cos(THI81);
386      V81 = Rf*sin(THI81);
387      W81 = Zf;
388      U82 = Rf*cos(THI82);
389      V82 = Rf*sin(THI82);
390      W82 = Zf;
391      U83 = Rf*cos(THI83);
392      V83 = Rf*sin(THI83);
393      W83 = Zf;
394      U84 = Rf*cos(THI84);
395      V84 = Rf*sin(THI84);
396      W84 = Zf;
397      U85 = Rf*cos(THI85);
398      V85 = Rf*sin(THI85);
399      W85 = Zf;
400      U86 = Rf*cos(THI86);
401      V86 = Rf*sin(THI86);
402      W86 = Zf;
403      U87 = Rf*cos(THI87);
404      V87 = Rf*sin(THI87);
405      W87 = Zf;
406      U88 = Rf*cos(THI88);
407      V88 = Rf*sin(THI88);
408      W88 = Zf;
409      U89 = Rf*cos(THI89);
410      V89 = Rf*sin(THI89);
411      W89 = Zf;
412      U90 = Rf*cos(THI90);
413      V90 = Rf*sin(THI90);
414      W90 = Zf;
415      U91 = Rf*cos(THI91);
416      V91 = Rf*sin(THI91);
417      W91 = Zf;
418      U92 = Rf*cos(THI92);
419      V92 = Rf*sin(THI92);
420      W92 = Zf;
421      U93 = Rf*cos(THI93);
422      V93 = Rf*sin(THI93);
423      W93 = Zf;
424      U94 = Rf*cos(THI94);
425      V94 = Rf*sin(THI94);
426      W94 = Zf;
427      U95 = Rf*cos(THI95);
428      V95 = Rf*sin(THI95);
429      W95 = Zf;
430      U96 = Rf*cos(THI96);
431      V96 = Rf*sin(THI96);
432      W96 = Zf;
433      U97 = Rf*cos(THI97);
434      V97 = Rf*sin(THI97);
435      W97 = Zf;
436      U98 = Rf*cos(THI98);
437      V98 = Rf*sin(THI98);
438      W98 = Zf;
439      U99 = Rf*cos(THI99);
440      V99 = Rf*sin(THI99);
441      W99 = Zf;
442      U100 = Rf*cos(THI100);
443      V100 = Rf*sin(THI100);
444      W100 = Zf;
445      U101 = Rf*cos(THI101);
446      V101 = Rf*sin(THI101);
447      W101 = Zf;
448      U102 = Rf*cos(THI102);
449      V102 = Rf*sin(THI102);
450      W102 = Zf;
451      U103 = Rf*cos(THI103);
452      V103 = Rf*sin(THI103);
453      W103 = Zf;
454      U104 = Rf*cos(THI104);
455      V104 = Rf*sin(THI104);
456      W104 = Zf;
457      U105 = Rf*cos(THI105);
458      V105 = Rf*sin(THI105);
459      W105 = Zf;
460      U106 = Rf*cos(THI106);
461      V106 = Rf*sin(THI106);
462      W106 = Zf;
463      U107 = Rf*cos(THI107);
464      V107 = Rf*sin(THI107);
465      W107 = Zf;
466      U108 = Rf*cos(THI108);
467      V108 = Rf*sin(THI108);
468      W108 = Zf;
469      U109 = Rf*cos(THI109);
470      V109 = Rf*sin(THI109);
471      W109 = Zf;
472      U110 = Rf*cos(THI110);
473      V110 = Rf*sin(THI110);
474      W110 = Zf;
475      U111 = Rf*cos(THI111);
476      V111 = Rf*sin(THI111);
477      W111 = Zf;
478      U112 = Rf*cos(THI112);
479      V112 = Rf*sin(THI112);
480      W112 = Zf;
481      U113 = Rf*cos(THI113);
482      V113 = Rf*sin(THI113);
483      W113 = Zf;
484      U114 = Rf*cos(THI114);
485      V114 = Rf*sin(THI114);
486      W114 = Zf;
487      U115 = Rf*cos(THI115);
488      V115 = Rf*sin(THI115);
489      W115 = Zf;
490      U116 = Rf*cos(THI116);
491      V116 = Rf*sin(THI116);
492      W116 = Zf;
493      U117 = Rf*cos(THI117);
494      V117 = Rf*sin(THI117);
495      W117 = Zf;
496      U118 = Rf*cos(THI118);
497      V118 = Rf*sin(THI118);
498      W118 = Zf;
499      U119 = Rf*cos(THI119);
500      V119 = Rf*sin(THI119);
501      W119 = Zf;
502      U120 = Rf*cos(THI120);
503      V120 = Rf*sin(THI120);
504      W120 = Zf;
505      U121 = Rf*cos(THI121);
506      V121 = Rf*sin(THI121);
507      W121 = Zf;
508      U122 = Rf*cos(THI122);
509      V122 = Rf*sin(THI122);
510      W122 = Zf;
511      U123 = Rf*cos(THI123);
512      V123 = Rf*sin(THI123);
513      W123 = Zf;
514      U124 = Rf*cos(THI124);
515      V124 = Rf*sin(THI124);
516      W124 = Zf;
517      U125 = Rf*cos(THI125);
518      V125 = Rf*sin(THI125);
519      W125 = Zf;
520      U126 = Rf*cos(THI126);
521      V126 = Rf*sin(THI126);
522      W126 = Zf;
523      U127 = Rf*cos(THI127);
524      V127 = Rf*sin(THI127);
525      W127 = Zf;
526      U128 = Rf*cos(THI128);
527      V128 = Rf*sin(THI128);
528      W128 = Zf;
529      U129 = Rf*cos(THI129);
530      V129 = Rf*sin(THI129);
531      W129 = Zf;
532      U130 = Rf*cos(THI130);
533      V130 = Rf*sin(THI130);
534      W130 = Zf;
535      U131 = Rf*cos(THI131);
536      V131 = Rf*sin(THI131);
537      W131 = Zf;
538      U132 = Rf*cos(THI132);
539      V132 = Rf*sin(THI132);
540      W132 = Zf;
541      U133 = Rf*cos(THI133);
542      V133 = Rf*sin(THI133);
543      W133 = Zf;
544      U134 = Rf*cos(THI134);
545      V134 = Rf*sin(THI134);
546      W134 = Zf;
547      U135 = Rf*cos(THI135);
548      V135 = Rf*sin(THI135);
549      W135 = Zf;
550      U136 = Rf*cos(THI136);
551      V136 = Rf*sin(THI136);
552      W136 = Zf;
553      U137 = Rf*cos(THI137);
554      V137 = Rf*sin(THI137);
555      W137 = Zf;
556      U138 = Rf*cos(THI138);
557      V138 = Rf*sin(THI138);
558      W138 = Zf;
559      U139 = Rf*cos(THI139);
560      V139 = Rf*sin(THI139);
561      W139 = Zf;
562      U140 = Rf*cos(THI140);
563      V140 = Rf*sin(THI140);
564      W140 = Zf;
565      U141 = Rf*cos(THI141);
566      V141 = Rf*sin(THI141);
567      W141 = Zf;
568      U142 = Rf*cos(THI142);
569      V142 = Rf*sin(THI142);
570      W142 = Zf;
571      U143 = Rf*cos(THI143);
572      V143 = Rf*sin(THI143);
573      W143 = Zf;
574      U144 = Rf*cos(THI144);
575      V144 = Rf*sin(THI144);
576      W144 = Zf;
577      U145 = Rf*cos(THI145);
578      V145 = Rf*sin(THI145);
579      W145 = Zf;
580      U146 = Rf*cos(THI146);
581      V146 = Rf*sin(THI146);
582      W146 = Zf;
583      U147 = Rf*cos(THI147);
584      V147 = Rf*sin(THI147);
585      W147 = Zf;
586      U148 = Rf*cos(THI148);
587      V148 = Rf*sin(THI148);
588      W148 = Zf;
589      U149 = Rf*cos(THI149);
590      V149 = Rf*sin(THI149);
591      W149 = Zf;
592      U150 = Rf*cos(THI150);
593      V150 = Rf*sin(THI150);
594      W150 = Zf;
595      U151 = Rf*cos(THI151);
596      V151 = Rf*sin(THI151);
597      W151 = Zf;
598      U152 = Rf*cos(THI152);
599      V152 = Rf*sin(THI152);
600      W152 = Zf;
601      U153 = Rf*cos(THI153);
602      V153 = Rf*sin(THI153);
603      W153 = Zf;
604      U154 = Rf*cos(THI154);
605      V154 = Rf*sin(THI154);
606      W154 = Zf;
607      U155 = Rf*cos(THI155);
608      V155 = Rf*sin(THI155);
609      W155 = Zf;
610      U156 = Rf*cos(THI156);
611      V156 = Rf*sin(THI156);
612      W156 = Zf;
613      U157 = Rf*cos(THI157);
614      V157 = Rf*sin(THI157);
615      W157 = Zf;
616      U158 = Rf*cos(THI158);
617      V158 = Rf*sin(THI158);
618      W158 = Zf;
619      U159 = Rf*cos(THI159);
620      V159 = Rf*sin(THI159);
621      W159 = Zf;
622      U160 = Rf*cos(THI160);
623      V160 = Rf*sin(THI160);
624      W160 = Zf;
625      U161 = Rf*cos(THI161);
626      V161 = Rf*sin(THI161);
627      W161 = Zf;
628      U162 = Rf*cos(THI162);
629      V162 = Rf*sin(THI162);
630      W162 = Zf;
631      U163 = Rf*cos(THI163);
632      V163 = Rf*sin(THI163);
633      W163 = Zf;
634      U164 = Rf*cos(THI164);
635      V164 = Rf*sin(THI164);
636      W164 = Zf;
637      U165 = Rf*cos(THI165);
638      V165 = Rf*sin(THI165);
639      W165 = Zf;
640      U166 = Rf*cos(THI166);
641      V166 = Rf*sin(THI166);
642      W166 = Zf;
643      U167 = Rf*cos(THI167);
644      V167 = Rf*sin(THI167);
645      W167 = Zf;
646      U168 = Rf*cos(THI168);
647      V168 = Rf*sin(THI168);
648      W168 = Zf;
649      U169 = Rf*cos(THI169);
650      V169 = Rf*sin(THI169);
651      W169 = Zf;
652      U170 = Rf*cos(THI170);
653      V170 = Rf*sin(THI170);
654      W170 = Zf;
655      U171 = Rf*cos(THI171);
656      V171 = Rf*sin(THI171);
657      W171 = Zf;
658      U172 = Rf*cos(THI172);
659      V172 = Rf*sin(THI172);
660      W172 = Zf;
661      U173 = Rf*cos(THI173);
662      V173 = Rf*sin(THI173);
663      W173 = Zf;
664      U174 = Rf*cos(THI174);
665      V174 = Rf*sin(THI174);
666      W174 = Zf;
667      U175 = Rf*cos(THI175);
668      V175 = Rf*sin(THI175);
669      W175 = Zf;
670      U176 = Rf*cos(THI176);
671      V176 = Rf*sin(THI176);
672      W176 = Zf;
673      U177 = Rf*cos(THI177);
674      V177 = Rf*sin(THI177);
675      W177 = Zf;
676      U178 = Rf*cos(THI178);
677      V178 = Rf*sin(THI178);
678      W178 = Zf;
679      U179 = Rf*cos(THI179);
680      V179 = Rf*sin(THI179);
681      W179 = Zf;
682      U180 = Rf*cos(THI180);
683      V180 = Rf*sin(THI180);
684      W180 = Zf;
685      U181 = Rf*cos(THI181);
686      V181 = Rf*sin(THI181);
687      W181 = Zf;
688      U182 = Rf*cos(THI182);
689      V182 = Rf*sin(THI182);
690      W182 = Zf;
691      U183 = Rf*cos(THI183);
692      V183 = Rf*sin(THI183);
693      W183 = Zf;
694      U184 = Rf*cos(THI184);
695      V184 = Rf*sin(THI184);
696      W184 = Zf;
697      U185 = Rf*cos(THI185);
698      V185 = Rf*sin(THI185);
699      W185 = Zf;
700      U186 = Rf*cos(THI186);
701      V186 = Rf*sin(THI186);
702      W186 = Zf;
703      U187 = Rf*cos(THI187);
704      V187 = Rf*sin(THI187);
705      W187 = Zf;
706      U188 = Rf*cos(THI188);
707      V188 = Rf*sin(THI188);
708      W188 = Zf;
709      U189 = Rf*cos(THI189);
710      V189 = Rf*sin(THI189);
711      W189 = Zf;
712      U190 = Rf*cos(THI190);
713      V190 = Rf*sin(THI190);
714      W190 = Zf;
715      U191 = Rf*cos(THI191);
716      V191 = Rf*sin(THI191);
717      W191 = Zf;
718      U192 = Rf*cos(THI192);
719      V192 = Rf*sin(THI192);
720      W192 = Zf;
721      U193 = Rf*cos(THI193);
722      V193 = Rf*sin(THI193);
723      W193 = Zf;
724      U194 = Rf*cos(THI194);
725      V194 = Rf*sin(THI194);
726      W194 = Zf;
727      U195 = Rf*cos(THI195);
728      V195 = Rf*sin(THI195);
729      W195 = Zf;
730      U196 = Rf*cos(THI196);
731      V196 = Rf*sin(THI196);
732      W196 = Zf;
733      U197 = Rf*cos(THI197);
734      V197 = Rf*sin(THI197);
735      W197 = Zf;
736      U198 = Rf*cos(THI198);
737      V198 = Rf*sin(THI198);
738      W198 = Zf;
739      U199 = Rf*cos(THI199);
740      V199 = Rf*sin(THI199);
741      W199 = Zf;
742      U200 = Rf*cos(THI200);
743      V200 = Rf*sin(THI200);
744      W200 = Zf;
745      U201 = Rf*cos(THI201);
746      V201 = Rf*sin(THI201);
747      W201 = Zf;
748      U202 = Rf*cos(THI202);
749      V202 = Rf*sin(THI202);
750      W202 = Zf;
751      U203 = Rf*cos(THI203);
752      V203 = Rf*sin(THI203);
753      W203 = Zf;
754      U204 = Rf*cos(THI204);
755      V204 = Rf*sin(THI204);
756      W204 = Zf;
757      U205 = Rf*cos(THI205);
758      V205 = Rf*sin(THI205);
759      W205 = Zf;
760      U206 = Rf*cos(THI206);
761      V206 = Rf*sin(THI206);
762      W206 = Zf;
763      U207 = Rf*cos(THI207);
764      V207 = Rf*sin(THI207);
765      W207 = Zf;
766      U208 = Rf*cos(THI208);
767      V208 = Rf*sin(THI208);
768      W208 = Zf;
769      U209 = Rf*cos(THI209);
770      V209 = Rf*sin(THI209);
771      W209 = Zf;
772      U210 = Rf*cos(THI210);
773      V210 = Rf*sin(THI210);
774      W210 = Zf;
775      U211 = Rf*cos(THI211);
776      V211 = Rf*sin(THI211);
777      W211 = Zf;
778      U212 = Rf*cos(THI212);
779      V212 = Rf*sin(THI212);
780      W212 = Zf;
781      U213 = Rf*cos(THI213);
782      V213 = Rf*sin(THI213);
783      W213 = Zf;
784      U214 = Rf*cos(THI214);
785      V214 = Rf*sin(THI214);
786      W214 = Zf;
787      U215 = Rf*cos(THI215);
788      V215 = Rf*sin(THI215);
789      W215 = Zf;
790      U216 = Rf*cos(THI216);
791      V216 = Rf*sin(THI216);
792      W216 = Zf;
793      U217 = Rf*cos(THI217);
794      V217 = Rf*sin(THI217);
795      W217 = Zf;
796      U218 = Rf*cos(THI218);
797      V218 = Rf*sin(THI218);
798      W218 = Zf;
799      U219 = Rf*cos(THI219);
800      V219 = Rf*sin(THI219);
801      W219 = Zf;
802      U220 = Rf*cos(THI220);
803      V220 = Rf*sin(THI220);
804      W220 = Zf;
805      U221 = Rf*cos(THI221);
806      V221 = Rf*sin(THI221);
807      W221 = Zf;
808      U222 = Rf*cos(THI222);
809      V222 = Rf*sin(THI222);
810      W222 = Zf;
811      U223 = Rf*cos(THI223);
812      V223 = Rf*sin(THI223);
813      W223 = Zf;
814      U224 = Rf*cos(THI224);
815      V224 = Rf*sin(THI224);
816      W224 = Zf;
817      U225 = Rf*cos(THI225);
818      V225 = Rf*sin(THI225);
819      W225 = Zf;
820      U226 = Rf*cos(THI226);
821      V226 = Rf*sin(THI226);
822      W226 = Zf;
823      U227 = Rf*cos(THI227);
824      V227 = Rf*sin(THI227);
825      W227 = Zf;
826      U228 = Rf*cos(THI228);
827      V228 = Rf*sin(THI228);
828      W228 = Zf;
829      U229 = Rf*cos(THI229);
830      V229 = Rf*sin(THI229);
831      W229 = Zf;
832      U230 = Rf*cos(THI230);
833      V230 = Rf*sin(THI230);
834      W230 = Zf;
835      U231 = Rf*cos(THI231);
836      V231 = Rf*sin(THI231);
837      W231 = Zf;
838      U232 = Rf*cos(THI232);
839      V232 = Rf*sin(THI232);
840      W232 = Zf;
841      U233 = Rf*cos(THI233);
842      V233 = Rf*sin(THI233);
843      W233 = Zf;
844      U234 = Rf*cos(THI234);
845      V234 = Rf*sin(THI234);
846      W234 = Zf;
847      U235 = Rf*cos(THI235);
848      V235 = Rf*sin(THI235);
849      W235 = Zf;
850      U236 = Rf*cos(THI236);
851      V236 = Rf*sin(THI236);
852      W236 = Zf;
853      U237 = Rf*cos(THI237);
854      V237 = Rf*sin(THI237);
855      W237 = Zf;
856      U238 = Rf*cos(THI238);
857      V238 = Rf*sin(THI238);
858      W238 = Zf;
859      U239 = Rf*cos(THI239);
860      V239 = Rf*sin(THI239);
861      W239 = Zf;
862      U240 =
```

```

1      if (dely == 0.0) /del/ == 0.0)
2          mel = 1;
3          g = 0;
4          /else
5              if (del/ == 0)
6                  mel = 1;
7              else
8                  mel = (dely/del);
9      }
10
11  /*****
12  *****/
13
14  /+
15
16  /+          COMPUTE DELTA THETA AND TOTAL TIME
17
18  /+
19
20  /+
21
22  /*****
23  *****/
24
25  if (TH1 - TH10)
26      DELTHT = TH1 - TH10;
27      TI = RS+sqrt(1-TAU)* (JA*(C+PI)*DELTHT) + ((DELTHT+
28  DELTHT)*(AAB));
29      /else
30      if (TH1 - TH10)
31          DELTHT = TH1 - TH10;
32          TI = RS+sqrt(1-TAU)* (JA*(C+PI)*DELTHT) + ((DELTHT+
33  DELTHT)*(AAB));
34      /else
35      TI = RS+sqrt(1-TAU)* (JA*(C+PI)*PI) + ((PI*PI)*AAB
36  ());
37  }
38
39  loopA, loopB, loopC;
40
41  return;
42
43  /*****
44  *****/
45
46  /+
47
48  /+          START LOOP AND COMPUTE THETA, I
49
50  /+
51
52  /*****
53  *****/
54
55  /+
56
57  /+
58
59  /+
60
61  /+
62
63  /+
64
65  /+
66
67  /+
68
69  /+
70
71  /+
72
73  /+
74
75  /+
76
77  /+
78
79  /+
80
81  /+
82
83  /+
84
85  /+
86
87  /+
88
89  /+
90
91  /+
92
93  /+
94
95  /+
96
97  /+
98
99  /+
100

```



```

005      00)
006      +-----+
007      +
008      +
009      +
010      +-----+
011      if(a == 0) /+ THETA
012      D = PI *
013      a = 0:
014      else(
015      if(a == 0)) /+ THETA
016      D = PI *
017      TH1 = TH1;
018      GAMMA = sqrt((P*R)+(Z*Z));
019      if(GAMMA > GAMMAX)
020      GAMMA = GAMMAX;
021      PSI = arctan(I/R);
022      ang3 = (1/(2*L1+L2))*(L1*L1)+(L2*L2) - (GAMMA*GAMMA
023      )
024      TH13 = arccos(ang3);
025      ang4 = (L2*sin(TH13))/GAMMA;
026      PHI = arctan(ang4);
027      TH12 = (PI/2) - (PSI-PHI);
028      )else( /+ THETA
029      D = PI *
030      GAMMA = sqrt((R-P)+(I-D));
031      if(GAMMA > GAMMAX)
032      GAMMA = GAMMAX;
033      PSI = arctan(I/R);
034      ang5 = (1/(2*L1+L2))*(L1*L1)+(L2*L2) - (GAMMA*GAMMA
035      )
036      TH13 = (2*PI) - arccos(ang5);
037      ang1 = (L2*sin(2*PI-TH13))/GAMMA;
038      PHI = arctan(ang1);
039      TH12 = (PI/2) - (PSI-PHI);
040
041      +-----+
042      +
043
044      +
045      +
046      +
047      +
048      +
049      +
050      +
051      +
052      +
053      +
054      +
055      +
056      +
057      +
058      +
059      +
060      +
061      +
062      +
063      +
064      +
065      +
066      +
067      +
068      +
069      +
070      +
071      +
072      +
073      +
074      +
075      +
076      +
077      +
078      +
079      +
080      +
081      +
082      +
083      +
084      +
085      +
086      +
087      +
088      +
089      +
090      +
091      +
092      +
093      +
094      +
095      +
096      +
097      +
098      +
099      +
100      +
101      +
102      +
103      +
104      +
105      +
106      +
107      +
108      +
109      +
110      +
111      +
112      +
113      +
114      +
115      +
116      +
117      +
118      +
119      +
120      +
121      +
122      +
123      +
124      +
125      +
126      +
127      +
128      +
129      +
130      +
131      +
132      +
133      +
134      +
135      +
136      +
137      +
138      +
139      +
140      +
141      +
142      +
143      +
144      +
145      +
146      +
147      +
148      +
149      +
150      +
151      +
152      +
153      +
154      +
155      +
156      +
157      +
158      +
159      +
160      +
161      +
162      +
163      +
164      +
165      +
166      +
167      +
168      +
169      +
170      +
171      +
172      +
173      +
174      +
175      +
176      +
177      +
178      +
179      +
180      +
181      +
182      +
183      +
184      +
185      +
186      +
187      +
188      +
189      +
190      +
191      +
192      +
193      +
194      +
195      +
196      +
197      +
198      +
199      +
200      +
201      +
202      +
203      +
204      +
205      +
206      +
207      +
208      +
209      +
210      +
211      +
212      +
213      +
214      +
215      +
216      +
217      +
218      +
219      +
220      +
221      +
222      +
223      +
224      +
225      +
226      +
227      +
228      +
229      +
230      +
231      +
232      +
233      +
234      +
235      +
236      +
237      +
238      +
239      +
240      +
241      +
242      +
243      +
244      +
245      +
246      +
247      +
248      +
249      +
250      +
251      +
252      +
253      +
254      +
255      +
256      +
257      +
258      +
259      +
260      +
261      +
262      +
263      +
264      +
265      +
266      +
267      +
268      +
269      +
270      +
271      +
272      +
273      +
274      +
275      +
276      +
277      +
278      +
279      +
280      +
281      +
282      +
283      +
284      +
285      +
286      +
287      +
288      +
289      +
290      +
291      +
292      +
293      +
294      +
295      +
296      +
297      +
298      +
299      +
300      +
301      +
302      +
303      +
304      +
305      +
306      +
307      +
308      +
309      +
310      +
311      +
312      +
313      +
314      +
315      +
316      +
317      +
318      +
319      +
320      +
321      +
322      +
323      +
324      +
325      +
326      +
327      +
328      +
329      +
330      +
331      +
332      +
333      +
334      +
335      +
336      +
337      +
338      +
339      +
340      +
341      +
342      +
343      +
344      +
345      +
346      +
347      +
348      +
349      +
350      +
351      +
352      +
353      +
354      +
355      +
356      +
357      +
358      +
359      +
360      +
361      +
362      +
363      +
364      +
365      +
366      +
367      +
368      +
369      +
370      +
371      +
372      +
373      +
374      +
375      +
376      +
377      +
378      +
379      +
380      +
381      +
382      +
383      +
384      +
385      +
386      +
387      +
388      +
389      +
390      +
391      +
392      +
393      +
394      +
395      +
396      +
397      +
398      +
399      +
400      +
401      +
402      +
403      +
404      +
405      +
406      +
407      +
408      +
409      +
410      +
411      +
412      +
413      +
414      +
415      +
416      +
417      +
418      +
419      +
420      +
421      +
422      +
423      +
424      +
425      +
426      +
427      +
428      +
429      +
430      +
431      +
432      +
433      +
434      +
435      +
436      +
437      +
438      +
439      +
440      +
441      +
442      +
443      +
444      +
445      +
446      +
447      +
448      +
449      +
450      +
451      +
452      +
453      +
454      +
455      +
456      +
457      +
458      +
459      +
460      +
461      +
462      +
463      +
464      +
465      +
466      +
467      +
468      +
469      +
470      +
471      +
472      +
473      +
474      +
475      +
476      +
477      +
478      +
479      +
480      +
481      +
482      +
483      +
484      +
485      +
486      +
487      +
488      +
489      +
490      +
491      +
492      +
493      +
494      +
495      +
496      +
497      +
498      +
499      +
500      +
501      +
502      +
503      +
504      +
505      +
506      +
507      +
508      +
509      +
510      +
511      +
512      +
513      +
514      +
515      +
516      +
517      +
518      +
519      +
520      +
521      +
522      +
523      +
524      +
525      +
526      +
527      +
528      +
529      +
530      +
531      +
532      +
533      +
534      +
535      +
536      +
537      +
538      +
539      +
540      +
541      +
542      +
543      +
544      +
545      +
546      +
547      +
548      +
549      +
550      +
551      +
552      +
553      +
554      +
555      +
556      +
557      +
558      +
559      +
560      +
561      +
562      +
563      +
564      +
565      +
566      +
567      +
568      +
569      +
570      +
571      +
572      +
573      +
574      +
575      +
576      +
577      +
578      +
579      +
580      +
581      +
582      +
583      +
584      +
585      +
586      +
587      +
588      +
589      +
590      +
591      +
592      +
593      +
594      +
595      +
596      +
597      +
598      +
599      +
600      +
601      +
602      +
603      +
604      +
605      +
606      +
607      +
608      +
609      +
610      +
611      +
612      +
613      +
614      +
615      +
616      +
617      +
618      +
619      +
620      +
621      +
622      +
623      +
624      +
625      +
626      +
627      +
628      +
629      +
630      +
631      +
632      +
633      +
634      +
635      +
636      +
637      +
638      +
639      +
640      +
641      +
642      +
643      +
644      +
645      +
646      +
647      +
648      +
649      +
650      +
651      +
652      +
653      +
654      +
655      +
656      +
657      +
658      +
659      +
660      +
661      +
662      +
663      +
664      +
665      +
666      +
667      +
668      +
669      +
670      +
671      +
672      +
673      +
674      +
675      +
676      +
677      +
678      +
679      +
680      +
681      +
682      +
683      +
684      +
685      +
686      +
687      +
688      +
689      +
690      +
691      +
692      +
693      +
694      +
695      +
696      +
697      +
698      +
699      +
700      +
701      +
702      +
703      +
704      +
705      +
706      +
707      +
708      +
709      +
710      +
711      +
712      +
713      +
714      +
715      +
716      +
717      +
718      +
719      +
720      +
721      +
722      +
723      +
724      +
725      +
726      +
727      +
728      +
729      +
730      +
731      +
732      +
733      +
734      +
735      +
736      +
737      +
738      +
739      +
740      +
741      +
742      +
743      +
744      +
745      +
746      +
747      +
748      +
749      +
750      +
751      +
752      +
753      +
754      +
755      +
756      +
757      +
758      +
759      +
760      +
761      +
762      +
763      +
764      +
765      +
766      +
767      +
768      +
769      +
770      +
771      +
772      +
773      +
774      +
775      +
776      +
777      +
778      +
779      +
780      +
781      +
782      +
783      +
784      +
785      +
786      +
787      +
788      +
789      +
790      +
791      +
792      +
793      +
794      +
795      +
796      +
797      +
798      +
799      +
800      +
801      +
802      +
803      +
804      +
805      +
806      +
807      +
808      +
809      +
810      +
811      +
812      +
813      +
814      +
815      +
816      +
817      +
818      +
819      +
820      +
821      +
822      +
823      +
824      +
825      +
826      +
827      +
828      +
829      +
830      +
831      +
832      +
833      +
834      +
835      +
836      +
837      +
838      +
839      +
840      +
841      +
842      +
843      +
844      +
845      +
846      +
847      +
848      +
849      +
850      +
851      +
852      +
853      +
854      +
855      +
856      +
857      +
858      +
859      +
860      +
861      +
862      +
863      +
864      +
865      +
866      +
867      +
868      +
869      +
870      +
871      +
872      +
873      +
874      +
875      +
876      +
877      +
878      +
879      +
880      +
881      +
882      +
883      +
884      +
885      +
886      +
887      +
888      +
889      +
890      +
891      +
892      +
893      +
894      +
895      +
896      +
897      +
898      +
899      +
900      +
901      +
902      +
903      +
904      +
905      +
906      +
907      +
908      +
909      +
910      +
911      +
912      +
913      +
914      +
915      +
916      +
917      +
918      +
919      +
920      +
921      +
922      +
923      +
924      +
925      +
926      +
927      +
928      +
929      +
930      +
931      +
932      +
933      +
934      +
935      +
936      +
937      +
938      +
939      +
940      +
941      +
942      +
943      +
944      +
945      +
946      +
947      +
948      +
949      +
950      +
951      +
952      +
953      +
954      +
955      +
956      +
957      +
958      +
959      +
960      +
961      +
962      +
963      +
964      +
965      +
966      +
967      +
968      +
969      +
970      +
971      +
972      +
973      +
974      +
975      +
976      +
977      +
978      +
979      +
980      +
981      +
982      +
983      +
984      +
985      +
986      +
987      +
988      +
989      +
990      +
991      +
992      +
993      +
994      +
995      +
996      +
997      +
998      +
999      +
1000      +

```

```

342     DEL4THT = THT4f - THT4i;
343     THT4 = THT4i + (DEL4THT * ((t/TT) - ((1/(2*PI)) * sin(((2*
PI)*t)/TT)))));
344     }else{
345     DEL4THT = THT4i - THT4f;
346     THT4 = THT4f + (DEL4THT * (1 - (t/TT) + ((1/(2*PI)) * sin(((2*
PI)*t)/TT)))));
347     }
348     if (THT5i < THT5f){
349     DEL5THT = THT5f - THT5i;
350     THT5 = THT5i + (DEL5THT * ((t/TT) - ((1/(2*PI)) * sin(((2*
PI)*t)/TT)))));
351     }else{
352     DEL5THT = THT5i - THT5f;
353     THT5 = THT5f + (DEL5THT * (1 - (t/TT) + ((1/(2*PI)) * sin(((2*
PI)*t)/TT)))));
354     }
355     /*-----*/
356     /*
357     *
358     *
359     *
360     *
361     *
362     *
363     *
364     *
365     *
366     *
367     *
368     *
369     *
370     *
371     *
372     *
373     *
374     *
375     *
376     *
377     *
378     *
379     *
380     *
381     *
382     *
383     *
384     *
385     *
386     *
387     *
388     *
389     *
390     *
391     *
392     *
393     *
394     *
395     *
396     *
397     *
398     *
399     *
400     *
401     *
402     *
403     *
404     *
405     *
406     *
407     *
408     *
409     *
410     *
411     *
412     *
413     *
414     *
415     *
416     *
417     *
418     *
419     *
420     *
421     *
422     *
423     *
424     *
425     *
426     *
427     *
428     *
429     *
430     *
431     *
432     *
433     *
434     *
435     *
436     *
437     *
438     *
439     *
440     *
441     *
442     *
443     *
444     *
445     *
446     *
447     *
448     *
449     *
450     *
451     *
452     *
453     *
454     *
455     *
456     *
457     *
458     *
459     *
460     *
461     *
462     *
463     *
464     *
465     *
466     *
467     *
468     *
469     *
470     *
471     *
472     *
473     *
474     *
475     *
476     *
477     *
478     *
479     *
480     *
481     *
482     *
483     *
484     *
485     *
486     *
487     *
488     *
489     *
490     *
491     *
492     *
493     *
494     *
495     *
496     *
497     *
498     *
499     *
500     *
501     *
502     *
503     *
504     *
505     *
506     *
507     *
508     *
509     *
510     *
511     *
512     *
513     *
514     *
515     *
516     *
517     *
518     *
519     *
520     *
521     *
522     *
523     *
524     *
525     *
526     *
527     *
528     *
529     *
530     *
531     *
532     *
533     *
534     *
535     *
536     *
537     *
538     *
539     *
540     *
541     *
542     *
543     *
544     *
545     *
546     *
547     *
548     *
549     *
550     *
551     *
552     *
553     *
554     *
555     *
556     *
557     *
558     *
559     *
560     *
561     *
562     *
563     *
564     *
565     *
566     *
567     *
568     *
569     *
570     *
571     *
572     *
573     *
574     *
575     *
576     *
577     *
578     *
579     *
580     *
581     *
582     *
583     *
584     *
585     *
586     *
587     *
588     *
589     *
590     *
591     *
592     *
593     *
594     *
595     *
596     *
597     *
598     *
599     *
600     *
601     *
602     *
603     *
604     *
605     *
606     *
607     *
608     *
609     *
610     *
611     *
612     *
613     *
614     *
615     *
616     *
617     *
618     *
619     *
620     *
621     *
622     *
623     *
624     *
625     *
626     *
627     *
628     *
629     *
630     *
631     *
632     *
633     *
634     *
635     *
636     *
637     *
638     *
639     *
640     *
641     *
642     *
643     *
644     *
645     *
646     *
647     *
648     *
649     *
650     *
651     *
652     *
653     *
654     *
655     *
656     *
657     *
658     *
659     *
660     *
661     *
662     *
663     *
664     *
665     *
666     *
667     *
668     *
669     *
670     *
671     *
672     *
673     *
674     *
675     *
676     *
677     *
678     *
679     *
680     *
681     *
682     *
683     *
684     *
685     *
686     *
687     *
688     *
689     *
690     *
691     *
692     *
693     *
694     *
695     *
696     *
697     *
698     *
699     *
700     *
701     *
702     *
703     *
704     *
705     *
706     *
707     *
708     *
709     *
710     *
711     *
712     *
713     *
714     *
715     *
716     *
717     *
718     *
719     *
720     *
721     *
722     *
723     *
724     *
725     *
726     *
727     *
728     *
729     *
730     *
731     *
732     *
733     *
734     *
735     *
736     *
737     *
738     *
739     *
740     *
741     *
742     *
743     *
744     *
745     *
746     *
747     *
748     *
749     *
750     *
751     *
752     *
753     *
754     *
755     *
756     *
757     *
758     *
759     *
760     *
761     *
762     *
763     *
764     *
765     *
766     *
767     *
768     *
769     *
770     *
771     *
772     *
773     *
774     *
775     *
776     *
777     *
778     *
779     *
780     *
781     *
782     *
783     *
784     *
785     *
786     *
787     *
788     *
789     *
790     *
791     *
792     *
793     *
794     *
795     *
796     *
797     *
798     *
799     *
800     *
801     *
802     *
803     *
804     *
805     *
806     *
807     *
808     *
809     *
810     *
811     *
812     *
813     *
814     *
815     *
816     *
817     *
818     *
819     *
820     *
821     *
822     *
823     *
824     *
825     *
826     *
827     *
828     *
829     *
830     *
831     *
832     *
833     *
834     *
835     *
836     *
837     *
838     *
839     *
840     *
841     *
842     *
843     *
844     *
845     *
846     *
847     *
848     *
849     *
850     *
851     *
852     *
853     *
854     *
855     *
856     *
857     *
858     *
859     *
860     *
861     *
862     *
863     *
864     *
865     *
866     *
867     *
868     *
869     *
870     *
871     *
872     *
873     *
874     *
875     *
876     *
877     *
878     *
879     *
880     *
881     *
882     *
883     *
884     *
885     *
886     *
887     *
888     *
889     *
890     *
891     *
892     *
893     *
894     *
895     *
896     *
897     *
898     *
899     *
900     *
901     *
902     *
903     *
904     *
905     *
906     *
907     *
908     *
909     *
910     *
911     *
912     *
913     *
914     *
915     *
916     *
917     *
918     *
919     *
920     *
921     *
922     *
923     *
924     *
925     *
926     *
927     *
928     *
929     *
930     *
931     *
932     *
933     *
934     *
935     *
936     *
937     *
938     *
939     *
940     *
941     *
942     *
943     *
944     *
945     *
946     *
947     *
948     *
949     *
950     *
951     *
952     *
953     *
954     *
955     *
956     *
957     *
958     *
959     *
960     *
961     *
962     *
963     *
964     *
965     *
966     *
967     *
968     *
969     *
970     *
971     *
972     *
973     *
974     *
975     *
976     *
977     *
978     *
979     *
980     *
981     *
982     *
983     *
984     *
985     *
986     *
987     *
988     *
989     *
990     *
991     *
992     *
993     *
994     *
995     *
996     *
997     *
998     *
999     *
1000    *

```

```

385     THTS = THTEMAA;
386     else(
387         if (THTS < THTEMIN)
388             THTS = THTEMIN;
389     )
390     /*****
391     */
392     /*      COMPUTE MOTOR ANGLES FROM THETA 4 & 5
393     */
394     /*****
395     */
396     PSIA = -THT4 - (AW*THTS);
397     PSIS = THT4 - (AW*THTS);
398     /*****
399     */
400     /*      CONVERT ANGLES TO INTEGERS
401     */
402     /*****
403     */
404     val001 = THT1;
405     val011 = THT2;
406     val123 = THT3;
407     val001 = PSIA;
408     val041 = PSIS;
409     cal001;
410     corang001 = corang001;
411     corang011 = corang011;
412     corang021 = corang021;
413     corang031 = corang031;
414     corang041 = corang041;
415     /*****
416     */
417     /*      SET THE AND OF ALL INPUTS TO THE CONTROL CUS
418     */
419     /*
420     */
421     /*****
422     */
423     /*      SET THE AND OF ALL INPUTS TO THE CONTROL CUS
424     */

```

```

422 memod[i] = (memod[i] & 0x0000000F) & 0x0000000F;
423 memod[i] = memod[i];
424 }
425 btree_invert_memod(memod,memod,i);
426 t = t + DELTA;
427 synchronize();
428 }
429 return;
430 }
431 scr_invert(10,0);
432 printf("a LOST SYNCHRONIZATION\n");
433 return;
434 }
435 double intang;
436 int intang[3];
437 {
438 double THT11,THT21,THT31,PS11,PS151;
439 /******
440 *
441 *
442 */
443 /******
444 if(intang[3] < 2655) && (intang[3] >= 2048)
445     intang[3] = 2655;
446 if(intang[3] > 1429) && (intang[3] <= 2047)
447     intang[3] = 1429;
448 if(intang[3] >= 2655) && (intang[3] <= 4096)
449     THT11 = (intang[3] - 2655) * .0010942503;
450 if(intang[3] <= 1429)
451     THT11 = 3.141592654 - ((1429 - intang[3]) * .00109
452 42503);
453 /******
454 *
455 *
456 */
457 THT21
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```



```

501      )
502      car(val)
503      double val[4];
504      {
505      double THT1,THT2,THT3,PS14,PS15;
506      THT1 = val[0];
507      THT2 = val[1];
508      THT3 = val[2];
509      PS14 = val[3];
510      PS15 = val[4];
511      /*****
512      */
513      /*
514      */
515      /*****
516      */
517      if (THT1 > 3.141592654)
518          THT1 = 3.141592654;
519      if (THT1 < 0.0)
520          THT1 = 0.0;
521      if ((THT1 >= 0.0) && (THT1 < 1.577908975))
522          crrang[3] = 2655 + (THT1 * 913.8676832);
523      if (crrang[3] > 4096)
524          crrang[3] = 4096;
525      if ((THT1 >= 1.577908975) && (THT1 <= 3.141592654))
526          crrang[3] = (THT1 - 1.577908975) * 913.8676832;
527      /*****
528      */
529      /*
530      */
531      /*****
532      */
533      if (THT2 < 1.3089969)
534          THT2 = 1.3089969;
535      if (THT2 > 2.6179938)
536          THT2 = 2.6179938;
537      if ((THT2 >= 1.3089969) && (THT2 < 1.659610476))
538          crrang[4] = (1.657226194 - THT2) * 1275.002118;
539      if (THT2 <= 1.659610476) && (THT2 >= 2.6179938)
540          crrang[4] = 4096 - (THT2 - 1.659610476) * 1275.002118;
541      /*****
542      */
543      /*
544      */
545      /*****
546      */
547      if (THT3 < 1.577908975)
548          THT3 = 1.577908975;
549      if (THT3 > 3.141592654)
550          THT3 = 3.141592654;
551      if ((THT3 >= 1.577908975) && (THT3 < 1.577908975))
552          crrang[5] = 0;
553      if ((THT3 >= 1.577908975) && (THT3 < 3.141592654))
554          crrang[5] = (THT3 - 1.577908975) * 913.8676832;
555      if (THT3 >= 3.141592654)
556          crrang[5] = 4096 - (THT3 - 3.141592654) * 913.8676832;
557      /*****
558      */
559      /*
560      */
561      /*****
562      */
563      if (PS14 < 0.0)
564          PS14 = 0.0;
565      if (PS14 > 1.0)
566          PS14 = 1.0;
567      if (PS14 < 0.0)
568          PS14 = 0.0;
569      if (PS14 > 1.0)
570          PS14 = 1.0;
571      if (PS14 < 0.0)
572          PS14 = 0.0;
573      if (PS14 > 1.0)
574          PS14 = 1.0;
575      if (PS14 < 0.0)
576          PS14 = 0.0;
577      if (PS14 > 1.0)
578          PS14 = 1.0;
579      if (PS14 < 0.0)
580          PS14 = 0.0;
581      if (PS14 > 1.0)
582          PS14 = 1.0;
583      if (PS14 < 0.0)
584          PS14 = 0.0;
585      if (PS14 > 1.0)
586          PS14 = 1.0;
587      if (PS14 < 0.0)
588          PS14 = 0.0;
589      if (PS14 > 1.0)
590          PS14 = 1.0;
591      if (PS14 < 0.0)
592          PS14 = 0.0;
593      if (PS14 > 1.0)
594          PS14 = 1.0;
595      if (PS14 < 0.0)
596          PS14 = 0.0;
597      if (PS14 > 1.0)
598          PS14 = 1.0;
599      if (PS14 < 0.0)
600          PS14 = 0.0;
601      if (PS14 > 1.0)
602          PS14 = 1.0;
603      if (PS14 < 0.0)
604          PS14 = 0.0;
605      if (PS14 > 1.0)
606          PS14 = 1.0;
607      if (PS14 < 0.0)
608          PS14 = 0.0;
609      if (PS14 > 1.0)
610          PS14 = 1.0;
611      if (PS14 < 0.0)
612          PS14 = 0.0;
613      if (PS14 > 1.0)
614          PS14 = 1.0;
615      if (PS14 < 0.0)
616          PS14 = 0.0;
617      if (PS14 > 1.0)
618          PS14 = 1.0;
619      if (PS14 < 0.0)
620          PS14 = 0.0;
621      if (PS14 > 1.0)
622          PS14 = 1.0;
623      if (PS14 < 0.0)
624          PS14 = 0.0;
625      if (PS14 > 1.0)
626          PS14 = 1.0;
627      if (PS14 < 0.0)
628          PS14 = 0.0;
629      if (PS14 > 1.0)
630          PS14 = 1.0;
631      if (PS14 < 0.0)
632          PS14 = 0.0;
633      if (PS14 > 1.0)
634          PS14 = 1.0;
635      if (PS14 < 0.0)
636          PS14 = 0.0;
637      if (PS14 > 1.0)
638          PS14 = 1.0;
639      if (PS14 < 0.0)
640          PS14 = 0.0;
641      if (PS14 > 1.0)
642          PS14 = 1.0;
643      if (PS14 < 0.0)
644          PS14 = 0.0;
645      if (PS14 > 1.0)
646          PS14 = 1.0;
647      if (PS14 < 0.0)
648          PS14 = 0.0;
649      if (PS14 > 1.0)
650          PS14 = 1.0;
651      if (PS14 < 0.0)
652          PS14 = 0.0;
653      if (PS14 > 1.0)
654          PS14 = 1.0;
655      if (PS14 < 0.0)
656          PS14 = 0.0;
657      if (PS14 > 1.0)
658          PS14 = 1.0;
659      if (PS14 < 0.0)
660          PS14 = 0.0;
661      if (PS14 > 1.0)
662          PS14 = 1.0;
663      if (PS14 < 0.0)
664          PS14 = 0.0;
665      if (PS14 > 1.0)
666          PS14 = 1.0;
667      if (PS14 < 0.0)
668          PS14 = 0.0;
669      if (PS14 > 1.0)
670          PS14 = 1.0;
671      if (PS14 < 0.0)
672          PS14 = 0.0;
673      if (PS14 > 1.0)
674          PS14 = 1.0;
675      if (PS14 < 0.0)
676          PS14 = 0.0;
677      if (PS14 > 1.0)
678          PS14 = 1.0;
679      if (PS14 < 0.0)
680          PS14 = 0.0;
681      if (PS14 > 1.0)
682          PS14 = 1.0;
683      if (PS14 < 0.0)
684          PS14 = 0.0;
685      if (PS14 > 1.0)
686          PS14 = 1.0;
687      if (PS14 < 0.0)
688          PS14 = 0.0;
689      if (PS14 > 1.0)
690          PS14 = 1.0;
691      if (PS14 < 0.0)
692          PS14 = 0.0;
693      if (PS14 > 1.0)
694          PS14 = 1.0;
695      if (PS14 < 0.0)
696          PS14 = 0.0;
697      if (PS14 > 1.0)
698          PS14 = 1.0;
699      if (PS14 < 0.0)
700          PS14 = 0.0;
701      if (PS14 > 1.0)
702          PS14 = 1.0;
703      if (PS14 < 0.0)
704          PS14 = 0.0;
705      if (PS14 > 1.0)
706          PS14 = 1.0;
707      if (PS14 < 0.0)
708          PS14 = 0.0;
709      if (PS14 > 1.0)
710          PS14 = 1.0;
711      if (PS14 < 0.0)
712          PS14 = 0.0;
713      if (PS14 > 1.0)
714          PS14 = 1.0;
715      if (PS14 < 0.0)
716          PS14 = 0.0;
717      if (PS14 > 1.0)
718          PS14 = 1.0;
719      if (PS14 < 0.0)
720          PS14 = 0.0;
721      if (PS14 > 1.0)
722          PS14 = 1.0;
723      if (PS14 < 0.0)
724          PS14 = 0.0;
725      if (PS14 > 1.0)
726          PS14 = 1.0;
727      if (PS14 < 0.0)
728          PS14 = 0.0;
729      if (PS14 > 1.0)
730          PS14 = 1.0;
731      if (PS14 < 0.0)
732          PS14 = 0.0;
733      if (PS14 > 1.0)
734          PS14 = 1.0;
735      if (PS14 < 0.0)
736          PS14 = 0.0;
737      if (PS14 > 1.0)
738          PS14 = 1.0;
739      if (PS14 < 0.0)
740          PS14 = 0.0;
741      if (PS14 > 1.0)
742          PS14 = 1.0;
743      if (PS14 < 0.0)
744          PS14 = 
```

```

537      crrang[21] = 267. + (-1*TH12 - 1.52359287) + 972.789423
538      9.;
539      if(crrang[21] < 4096)
540      crrang[21] = 4096;
541      1+(-TH13 - 1.176752136) % (TH13 - 3.2504578);
542      crrang[21] = (TH13 - 1.776752136) * 972.7894239;
543      /*****
544      *****/
545      /*
546      */
547      /*****
548      *****/
549      if(PB14 < -4.765796055)
550      PB14 = -4.765796055;
551      if(PB14 > 1.624203402)
552      PB14 = 1.624203402;
553      1+(-PB14) % (-4.765796055) % (PB14 - 1.405492402);
554      crrang[0] = 2176 + ((PB14 + 4.765796055) * 571.674
555      54);
556      if(crrang[0] < 4096)
557      crrang[0] = 4096;
558      1+((PB14) % -1.405492402) % (PB14 < 1.624203402);
559      crrang[0] = (PB14 + 1.405492402) * 571.67454;
560      /*****
561      *****/
562      /*
563      */
564      /*****
565      *****/
566      if(PB15 < -1.624203402)
567      PB15 = -1.624203402;
568      if(PB15 > 4.765796055)
569      PB15 = 4.765796055;
570      1+PB15 % -1.624203402) % (PB15 < 1.316597585);
571      crrang[11] = 2160 + ((PB15 + 1.624203402) * 558.651
572      069);
573      if(crrang[11] < 4096)
574      crrang[11] = 4096;
575      1+PB15 % -1.316597585) % (PB15 > 4.765796055);
576      crrang[11] = (PB15 - 1.316597585) * 558.6510649;
577      /*****
578      *****/
579      /*****
580      *****/
581      /*****
582      *****/
583      /*****
584      *****/
585      /*****
586      *****/
587      /*****
588      *****/
589      /*****
590      *****/
591      /*****
592      *****/
593      /*****
594      *****/
595      /*****
596      *****/
597      /*****
598      *****/
599      /*****
600      *****/
601      /*****
602      *****/
603      /*****
604      *****/
605      /*****
606      *****/
607      /*****
608      *****/
609      /*****
610      *****/
611      /*****
612      *****/
613      /*****
614      *****/
615      /*****
616      *****/
617      /*****
618      *****/
619      /*****
620      *****/
621      /*****
622      *****/
623      /*****
624      *****/
625      /*****
626      *****/
627      /*****
628      *****/
629      /*****
630      *****/
631      /*****
632      *****/
633      /*****
634      *****/
635      /*****
636      *****/
637      /*****
638      *****/
639      /*****
640      *****/
641      /*****
642      *****/
643      /*****
644      *****/
645      /*****
646      *****/
647      /*****
648      *****/
649      /*****
650      *****/
651      /*****
652      *****/
653      /*****
654      *****/
655      /*****
656      *****/
657      /*****
658      *****/
659      /*****
660      *****/
661      /*****
662      *****/
663      /*****
664      *****/
665      /*****
666      *****/
667      /*****
668      *****/
669      /*****
670      *****/
671      /*****
672      *****/
673      /*****
674      *****/
675      /*****
676      *****/
677      /*****
678      *****/
679      /*****
680      *****/
681      /*****
682      *****/
683      /*****
684      *****/
685      /*****
686      *****/
687      /*****
688      *****/
689      /*****
690      *****/
691      /*****
692      *****/
693      /*****
694      *****/
695      /*****
696      *****/
697      /*****
698      *****/
699      /*****
700      *****/
701      /*****
702      *****/
703      /*****
704      *****/
705      /*****
706      *****/
707      /*****
708      *****/
709      /*****
710      *****/
711      /*****
712      *****/
713      /*****
714      *****/
715      /*****
716      *****/
717      /*****
718      *****/
719      /*****
720      *****/
721      /*****
722      *****/
723      /*****
724      *****/
725      /*****
726      *****/
727      /*****
728      *****/
729      /*****
730      *****/
731      /*****
732      *****/
733      /*****
734      *****/
735      /*****
736      *****/
737      /*****
738      *****/
739      /*****
740      *****/
741      /*****
742      *****/
743      /*****
744      *****/
745      /*****
746      *****/
747      /*****
748      *****/
749      /*****
750      *****/
751      /*****
752      *****/
753      /*****
754      *****/
755      /*****
756      *****/
757      /*****
758      *****/
759      /*****
760      *****/
761      /*****
762      *****/
763      /*****
764      *****/
765      /*****
766      *****/
767      /*****
768      *****/
769      /*****
770      *****/
771      /*****
772      *****/
773      /*****
774      *****/
775      /*****
776      *****/
777      /*****
778      *****/
779      /*****
780      *****/
781      /*****
782      *****/
783      /*****
784      *****/
785      /*****
786      *****/
787      /*****
788      *****/
789      /*****
790      *****/
791      /*****
792      *****/
793      /*****
794      *****/
795      /*****
796      *****/
797      /*****
798      *****/
799      /*****
800      *****/
801      /*****
802      *****/
803      /*****
804      *****/
805      /*****
806      *****/
807      /*****
808      *****/
809      /*****
810      *****/
811      /*****
812      *****/
813      /*****
814      *****/
815      /*****
816      *****/
817      /*****
818      *****/
819      /*****
820      *****/
821      /*****
822      *****/
823      /*****
824      *****/
825      /*****
826      *****/
827      /*****
828      *****/
829      /*****
830      *****/
831      /*****
832      *****/
833      /*****
834      *****/
835      /*****
836      *****/
837      /*****
838      *****/
839      /*****
840      *****/
841      /*****
842      *****/
843      /*****
844      *****/
845      /*****
846      *****/
847      /*****
848      *****/
849      /*****
850      *****/
851      /*****
852      *****/
853      /*****
854      *****/
855      /*****
856      *****/
857      /*****
858      *****/
859      /*****
860      *****/
861      /*****
862      *****/
863      /*****
864      *****/
865      /*****
866      *****/
867      /*****
868      *****/
869      /*****
870      *****/
871      /*****
872      *****/
873      /*****
874      *****/
875      /*****
876      *****/
877      /*****
878      *****/
879      /*****
880      *****/
881      /*****
882      *****/
883      /*****
884      *****/
885      /*****
886      *****/
887      /*****
888      *****/
889      /*****
890      *****/
891      /*****
892      *****/
893      /*****
894      *****/
895      /*****
896      *****/
897      /*****
898      *****/
899      /*****
900      *****/
901      /*****
902      *****/
903      /*****
904      *****/
905      /*****
906      *****/
907      /*****
908      *****/
909      /*****
910      *****/
911      /*****
912      *****/
913      /*****
914      *****/
915      /*****
916      *****/
917      /*****
918      *****/
919      /*****
920      *****/
921      /*****
922      *****/
923      /*****
924      *****/
925      /*****
926      *****/
927      /*****
928      *****/
929      /*****
930      *****/
931      /*****
932      *****/
933      /*****
934      *****/
935      /*****
936      *****/
937      /*****
938      *****/
939      /*****
940      *****/
941      /*****
942      *****/
943      /*****
944      *****/
945      /*****
946      *****/
947      /*****
948      *****/
949      /*****
950      *****/
951      /*****
952      *****/
953      /*****
954      *****/
955      /*****
956      *****/
957      /*****
958      *****/
959      /*****
960      *****/
961      /*****
962      *****/
963      /*****
964      *****/
965      /*****
966      *****/
967      /*****
968      *****/
969      /*****
970      *****/
971      /*****
972      *****/
973      /*****
974      *****/
975      /*****
976      *****/
977      /*****
978      *****/
979      /*****
980      *****/
981      /*****
982      *****/
983      /*****
984      *****/
985      /*****
986      *****/
987      /*****
988      *****/
989      /*****
990      *****/
991      /*****
992      *****/
993      /*****
994      *****/
995      /*****
996      *****/
997      /*****
998      *****/
999      /*****
1000     *****/

```



```

551  /******
552  *****
553  double arcsin(x)
554  double z;
555  {
556  double r,w,ang,y,zt,ang1,ang2,ang3,tc;
557  if ((fabs(x))>1) {
558      printf("ERROR IN COMPUTING ARGUMENT FOR ARCSINE'x"
559      );
560      ang = 0.0;
561      }else{
562      if ((fabs(x)) <= .707107) {
563          z = x;
564          ang2 = x + ((x*x*x)/6) + ((-x*x*x*x)/24) + .07514
565          ang3 = ang2 + ((x*x*x*x*x)/120) + .044640961;
566          ang = ang3 + ((x*x*x*x*x*x)/5040) + .00038194 ;
567      }else{
568          tc = (x*0.7-1);
569          z = 1 - (x*x);
570          if (z == 0)
571              ang = tc*(3.1415927/2);
572          else{
573              w = 1/2;
574              zt = 0;
575              y = .5+((x/w)-w);
576              while ((fabs(y)) > .00000001) { y = zt+y;
577                  w = w+y;
578                  zt = y;
579                  y = .5*(x/w)-w;
580              }
581              x = w;
582              ang2 = x + ((x*x*x)/6) + ((-x*x*x*x)/24) + .07514
583              ang3 = ang2 + ((x*x*x*x*x)/120) + .044640961;
584              ang1 = ang3 + ((x*x*x*x*x*x)/5040) + .00038194 ;
585              ang = tc*(1.570796-ang1);
586          }
587          return(ang);
588      }
589  }
590  /******
591  *****
592  double arctan(x)
593  double z;
594  {
595  double ang;
596  if (fabs(x) < 1)
597      printf("ERROR IN COMPUTING ARGUMENT FOR ARCTAN'x"
598      );
599      ang = 0.0;
600      }else{
601          ang = 1.570796 - arctan(1/x);
602      }
603      return(ang);
604  }
605  /******
606  *****

```

```

019      }
020      /***** **** */
021      *****/
022      double arctan(s)
023      double s;
024      {
025      double tz, cz, az, b1, b2, ang;
026      tz = (s > 0)?1:-1;
027      s = fabs(s);
028      cz = 0;
029      if(s > 1){
030          cz = 1;
031          s = 1/s;
032          :
033      az = s*s;
034      b1 = (-.00286523*az+.0161657)*az+.0429096)*az;
035      b2 = (-.01(b1+.075289)*az+.106863)*az+.142089)*az+.19963
036      az+az;
037      az = (b2 + .333333)*az+1)*az;
038      ang = (cz == 1)?(1.570796-az):tz*az;
039      return(ang);
040      }
041

```

END

FILMED

8-85

DTIC